US011475141B1

(12) **United States Patent**
Stewart

(10) **Patent No.: US 11,475,141 B1**
(45) **Date of Patent: Oct. 18, 2022**

(54) **APPARATUS AND METHODS FOR VERIFYING LOST USER DATA**

(71) Applicant: **MY JOB MATCHER, INC.**, Austin, TX (US)

(72) Inventor: **Arran Stewart**, Austin, TX (US)

(73) Assignee: **MY JOB MATCHER, INC.**, Austin, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **17/695,049**

(22) Filed: **Mar. 15, 2022**

(51) **Int. Cl.**
G06F 21/60 (2013.01)
G06F 11/14 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 21/602*** (2013.01); ***G06F 11/1469*** (2013.01); *G06F 2221/2107* (2013.01)

(58) **Field of Classification Search**
CPC G06F 21/602; G06F 11/1469; G06F 2221/2107
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

9,992,022	B1	6/2018	Chapman et al.	
11,068,908	B1	7/2021	Wong et al.	
11,170,346	B2	11/2021	Tummuru et al.	
2007/0156792	A1*	7/2007	D'Souza	G06F 11/2041
2007/0226535	A1*	9/2007	Gokhale	G06F 16/2365 714/6.12
2019/0130443	A1*	5/2019	Antzoulis	G06Q 50/01
2020/0019616	A1	1/2020	Sukhija et al.	
2020/0220726	A1	7/2020	Lougheed, III et al.	
2021/0173945	A1*	6/2021	Karr	G06F 3/065
2021/0314140	A1*	10/2021	Stephenson	H04L 63/0428
2022/0057519	A1*	2/2022	Goldstein	G01S 17/894
2022/0067738	A1*	3/2022	Fang	G06Q 20/407

FOREIGN PATENT DOCUMENTS

KR	102051231 B1	1/2019	
WO	WO-2013044726 A1 *	4/2013	G06F 11/1435

OTHER PUBLICATIONS

IBM, Blockchain for digital identity and credentials, Dec. 2, 2021.
Aaron Rose, Using Blockchain to Verify a Candidate's Credentials and Fill Job Vacancies, Nov. 11, 2017.

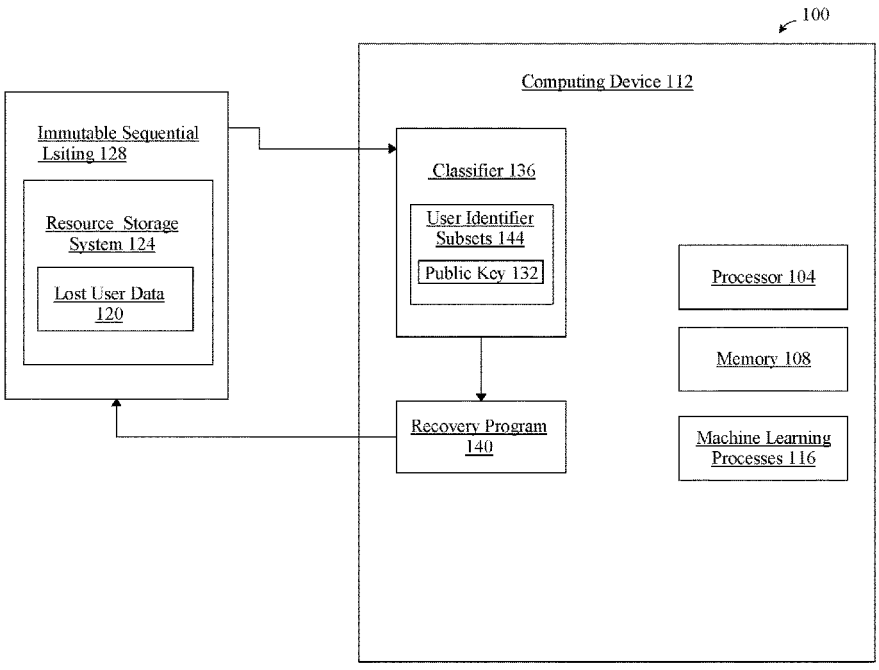
* cited by examiner

Primary Examiner — Meng Li
(74) *Attorney, Agent, or Firm* — Caldwell Intellectual Property Law

(57) **ABSTRACT**

Aspects relate to apparatuses and methods for using machine learning to verify lost user data in a resource data storage system. An exemplary apparatus includes a processor and a memory communicatively connected to the processor, the memory containing instructions configuring the processor to identify a plurality of lost user data stored in a resource data storage system potentially associated with a particular user of a plurality of users, verify, using a recovery program, the plurality of lost user data potentially associated with the particular user and link, as a function of the verification, the plurality of lost user data to the particular user.

20 Claims, 9 Drawing Sheets



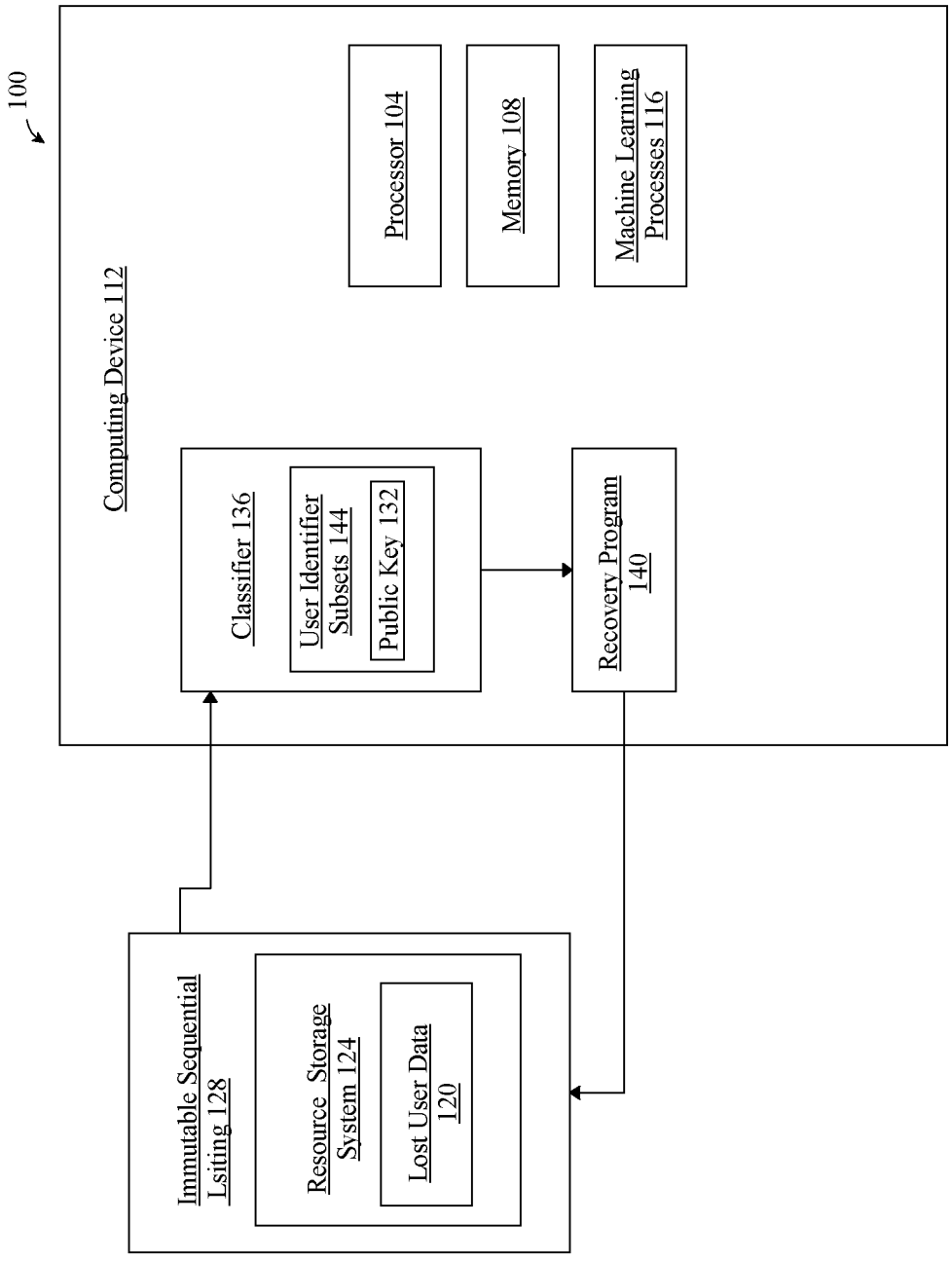

100
Computing Device 112
Processor 104
Memory 108
Machine Learning Processes 116
Classifier 136
User Identifier Subsets 144
Public Key 132
Recovery Program 140
Immutable Sequential Lsiting 128
Resource Storage System 124
Lost User Data 120

FIG. 1

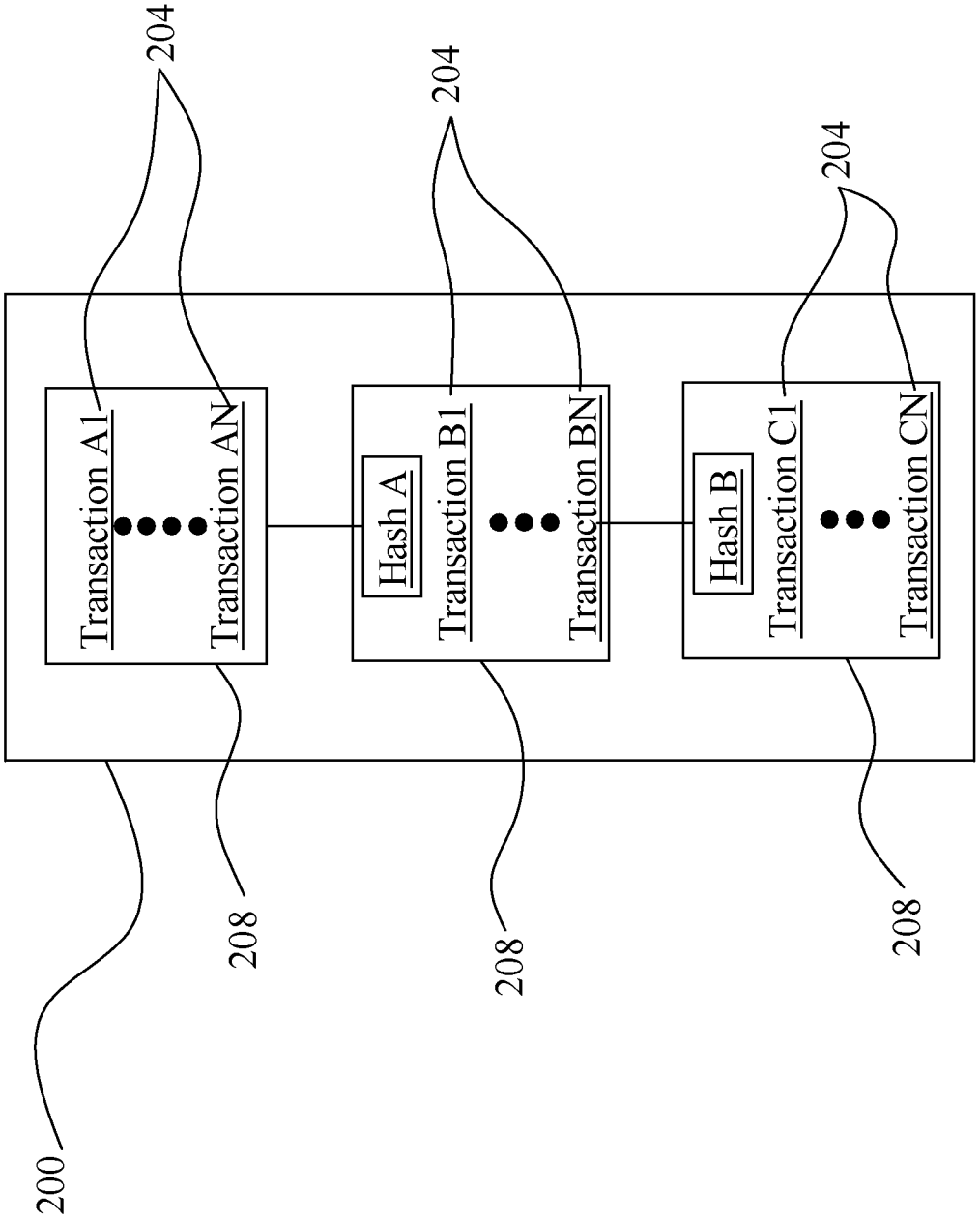

204
204
204
Transaction A1
Transaction AN
Hash A
Transaction B1
Transaction BN
Hash B
Transaction C1
Transaction CN
208
208
208
200

FIG. 2

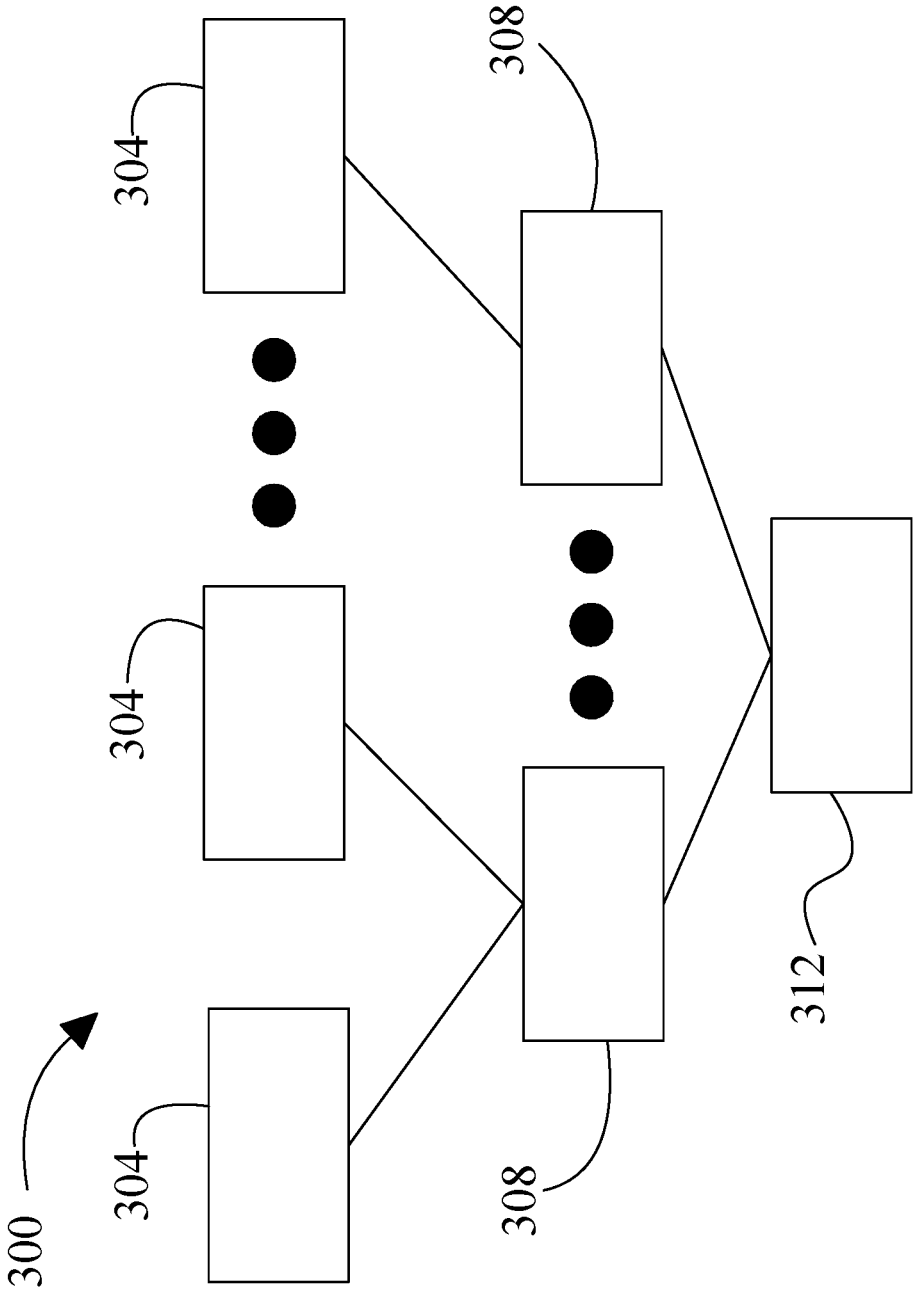


FIG. 3

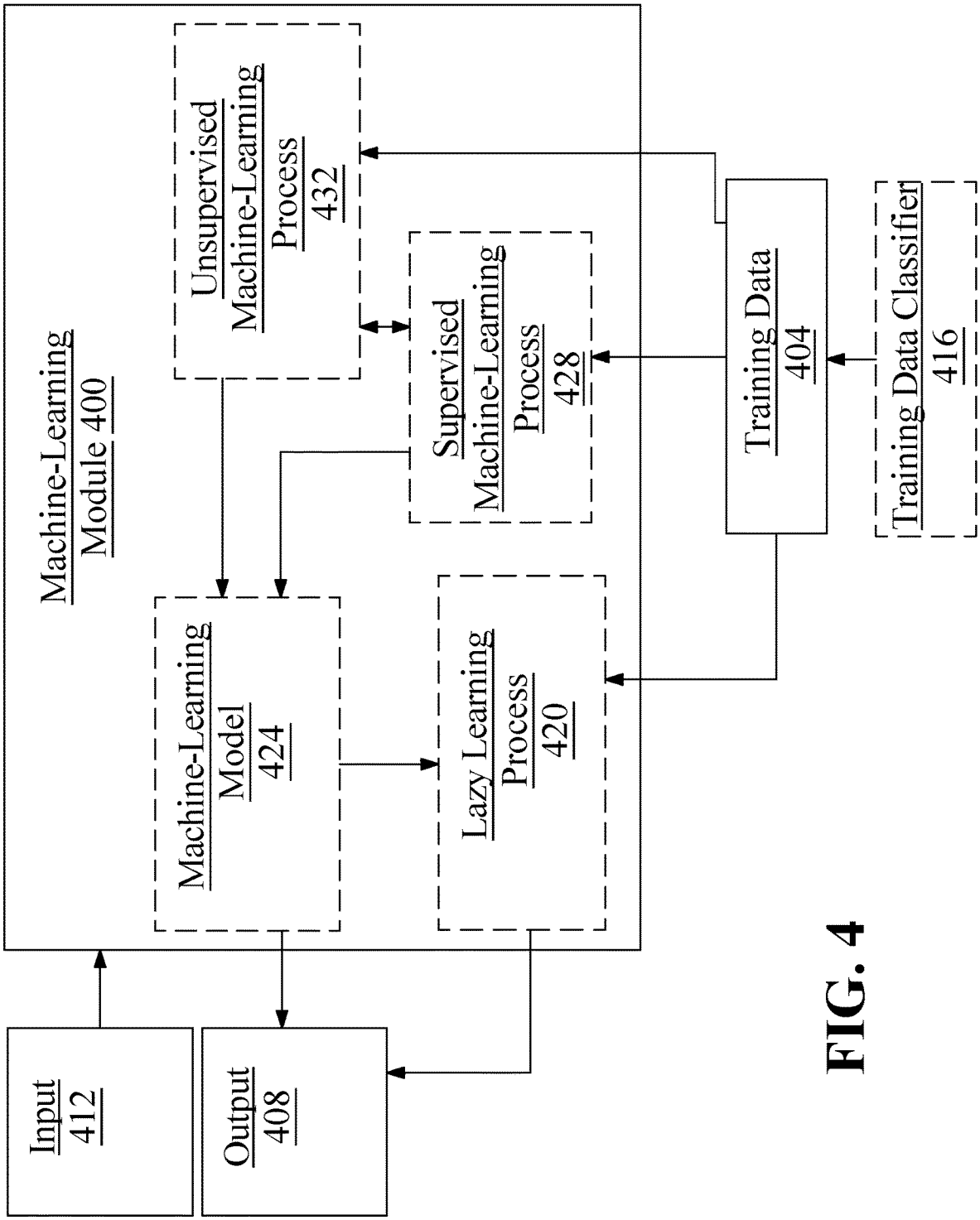

Machine-Learning Module 400
Unsupervised Machine-Learning Process 432
Supervised Machine-Learning Process 428
Machine-Learning Model 424
Lazy Learning Process 420
Training Data 404
Training Data Classifier 416
Input 412
Output 408

FIG. 4

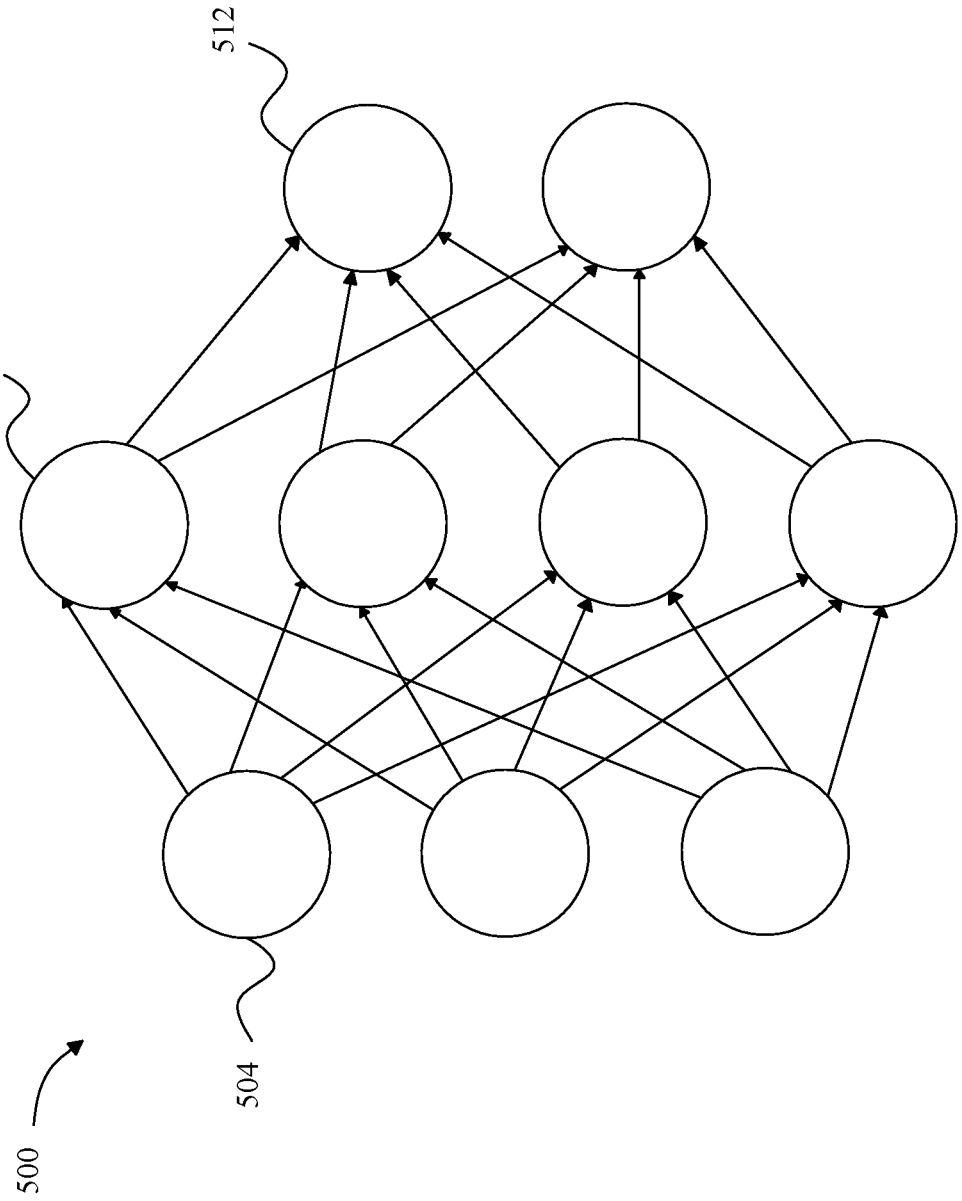


FIG. 5

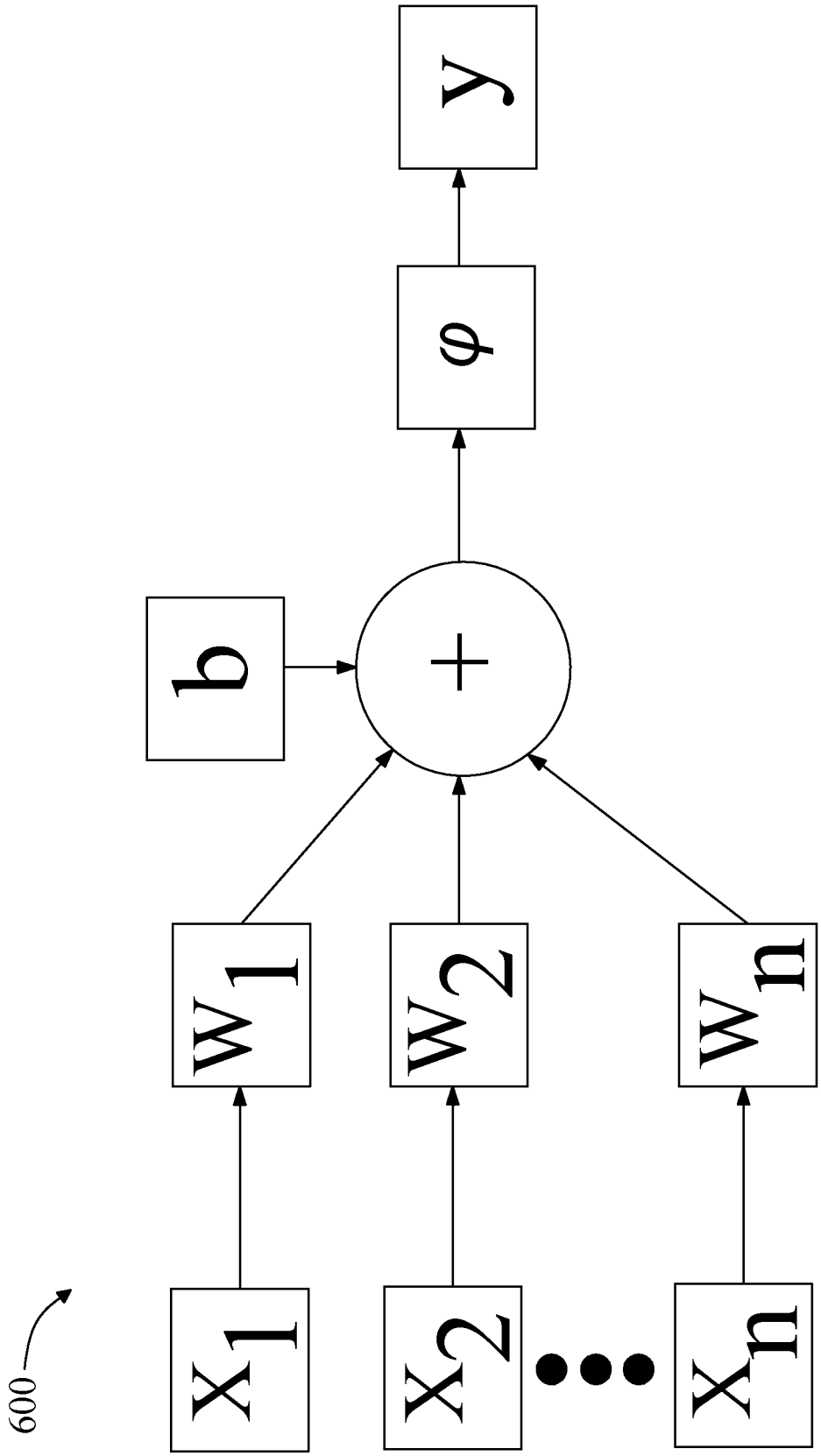


FIG. 6

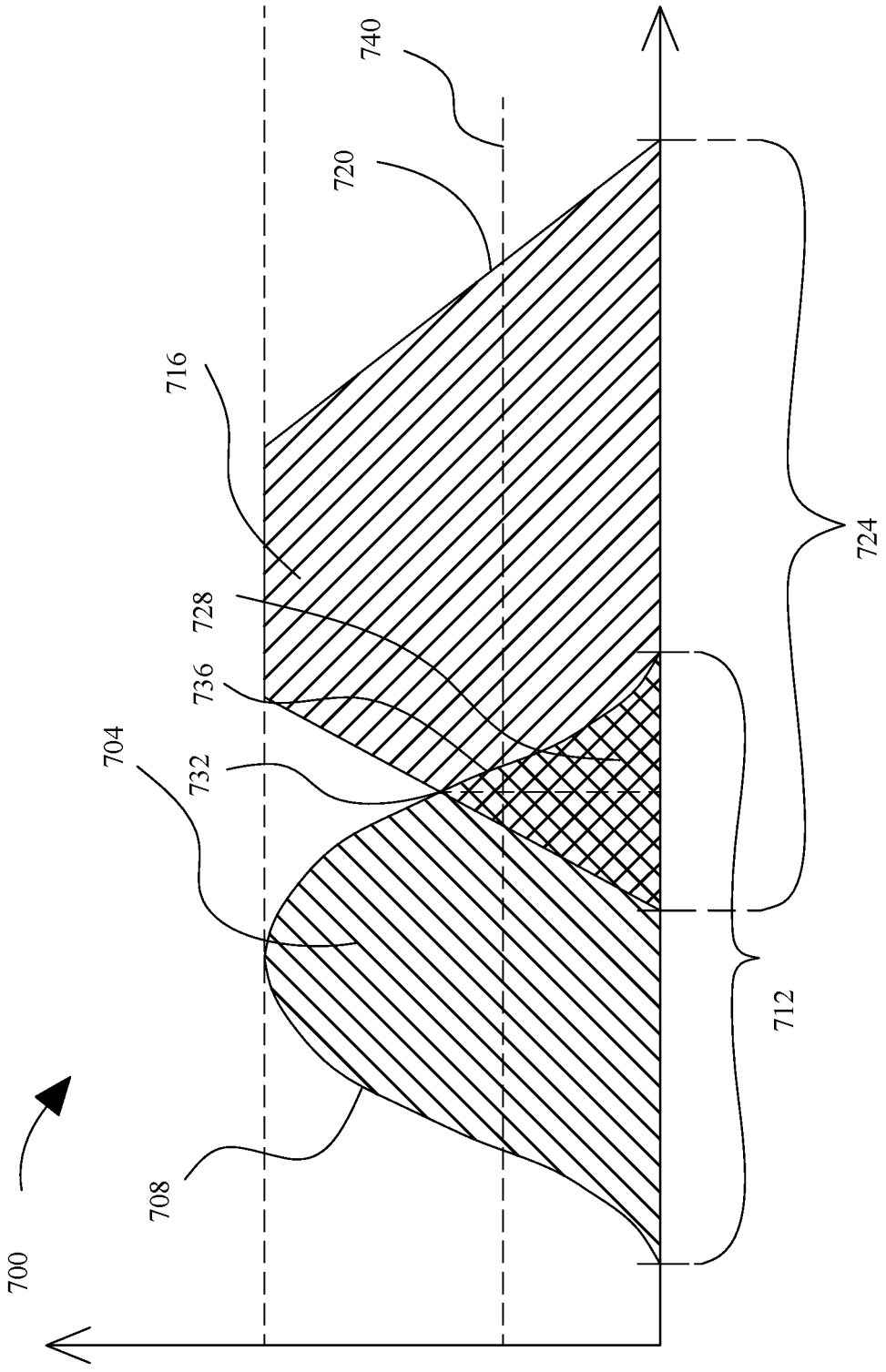


FIG. 7

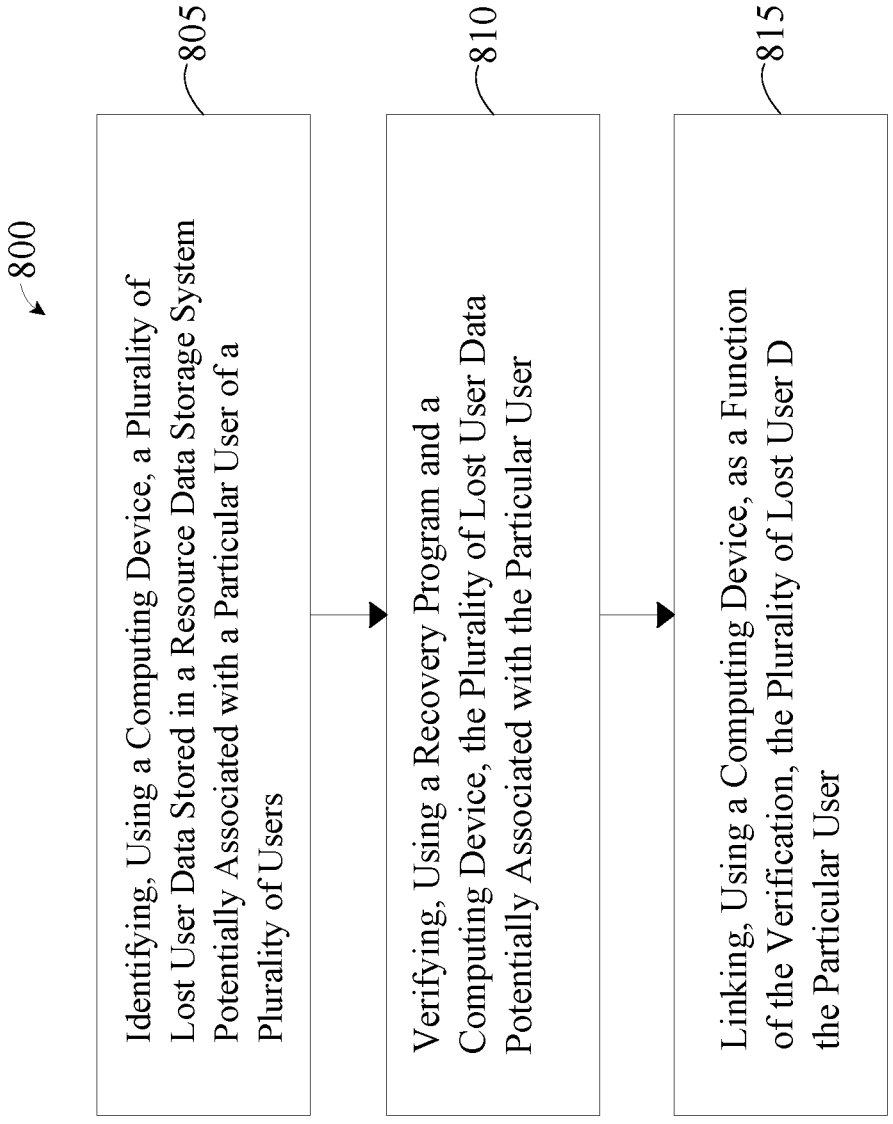

800
805
Identifying, Using a Computing Device, a Plurality of Lost User Data Stored in a Resource Data Storage System Potentially Associated with a Particular User of a Plurality of Users
810
Verifying, Using a Recovery Program and a Computing Device, the Plurality of Lost User Data Potentially Associated with the Particular User
815
Linking, Using a Computing Device, as a Function of the Verification, the Plurality of Lost User D the Particular User

FIG. 8

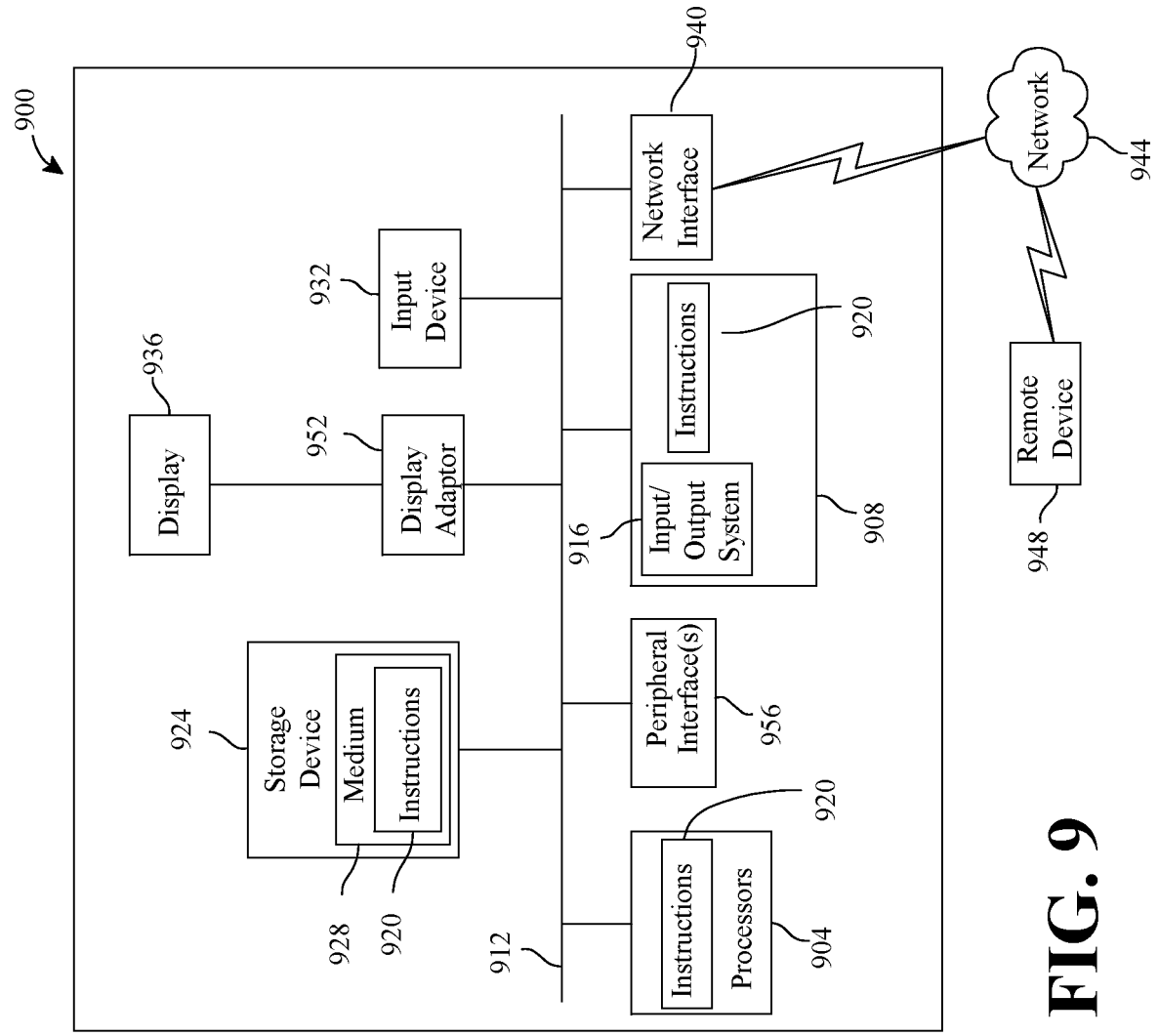

900
Display
936
952
Display Adaptor
932
Input Device
916
Input/ Output System
Instructions
920
908
940
Network Interface
Network
944
Remote Device
948
924
Storage Device
Medium
Instructions
928
920
912
Peripheral Interface(s)
956
Instructions
Processors
904
920

FIG. 9

APPARATUS AND METHODS FOR VERIFYING LOST USER DATA

FIELD OF THE INVENTION

The present invention generally relates to the field of resource storage systems. In particular, the present invention is directed to identifying and verifying lost user data in a resource storage system.

BACKGROUND

Resource data storages systems are a wonderful way to securely upload and store sensitive information. However, a maintenance problem arises when information is not verified effectively.

SUMMARY OF THE DISCLOSURE

In an aspect, an apparatus for using machine learning to verify lost user data in a resource data storage system is illustrated. The apparatus including at least a processor and a memory communicatively connected to the processor, the memory containing instructions configuring the processor to identify a plurality of lost user data stored in a resource data storage system potentially associated with a particular user of a plurality of users, verify, using a recovery program, the plurality of lost user data potentially associated with the particular user and link, as a function of the verification, the plurality of lost user data to the particular user.

In another aspect, a method for using machine learning to verify lost user data in a resource data storage system. The method includes using a computing device configured to identify a plurality of lost user data stored in a resource data storage system potentially associated with a particular user of a plurality of users, verify, using a recovery program, the plurality of lost user data potentially associated with the particular user and link, as a function of the verification, the plurality of lost user data to the particular user.

These and other aspects and features of non-limiting embodiments of the present invention will become apparent to those skilled in the art upon review of the following description of specific non-limiting embodiments of the invention in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

For the purpose of illustrating the invention, the drawings show aspects of one or more embodiments of the invention. However, it should be understood that the present invention is not limited to the precise arrangements and instrumentalities shown in the drawings, wherein: FIG. **1** is a block diagram illustrating an exemplary embodiment of an apparatus for using machine learning to verify lost user data; FIG. **2** is a block diagram illustrating an exemplary embodiment of an immutable sequential listing; FIG. **3** is a block diagram illustrating an exemplary embodiment of a cryptographic accumulator; FIG. **4** is a block diagram of an exemplary embodiment of a machine-learning module; FIG. **5** is a block diagram illustrating an exemplary embodiment of a neural network; FIG. **6** is a diagram of an exemplary embodiment of a node of a neural network; FIG. **7** is a diagram of an exemplary embodiment of a fuzzy set comparison; FIG. **8** is a flow diagram illustrating an exemplary embodiment of a method for using a computing device to verify lost user data; and FIG. **9** is a block diagram of a computing system that can be used to implement any one or more of the methodologies disclosed herein and any one or more portions thereof.

The drawings are not necessarily to scale and may be illustrated by phantom lines, diagrammatic representations and fragmentary views. In certain instances, details that are not necessary for an understanding of the embodiments or that render other details difficult to perceive may have been omitted.

DETAILED DESCRIPTION

At a high level, aspects of the present disclosure are directed to apparatuses and methods for using machine learning to verify lost user data in a resource data storage system. In an embodiment, lost user data be a job-seekers work experience.

Aspects of the present disclosure can be used to verify a plurality of lost user data of job seekers on job matching websites. Aspects of the present disclosure can also be used to add recovered lost user data to a job matching website immutable sequential listing. Apparatuses and methods for verifying lost user data may allow job matching services to work more fluidly.

Exemplary embodiments illustrating aspects of the present disclosure are described below in the context of several specific examples.

In an embodiment, methods and apparatuses described herein may perform or implement one or more aspects of a cryptographic system. In one embodiment, a cryptographic system is a system that converts data from a first form, known as "plaintext," which is intelligible when viewed in its intended format, into a second form, known as "ciphertext," which is not intelligible when viewed in the same way. Ciphertext may be unintelligible in any format unless first converted back to plaintext. In one embodiment, a process of converting plaintext into ciphertext is known as "encryption." Encryption process may involve the use of a datum, known as an "encryption key," to alter plaintext. Cryptographic system may also convert ciphertext back into plaintext, which is a process known as "decryption." Decryption process may involve the use of a datum, known as a "decryption key," to return the ciphertext to its original plaintext form. In embodiments of cryptographic systems that are "symmetric," decryption key is essentially the same as encryption key: possession of either key makes it possible to deduce the other key quickly without further secret knowledge. Encryption and decryption keys in symmetric cryptographic systems may be kept secret and shared only with persons or entities that the user of the cryptographic system wishes to be able to decrypt the ciphertext. One example of a symmetric cryptographic system is the Advanced Encryption Standard ("AES"), which arranges plaintext into matrices and then modifies the matrices through repeated permutations and arithmetic operations with an encryption key.

In embodiments of cryptographic systems that are "asymmetric," either encryption or decryption key cannot be readily deduced without additional secret knowledge, even given the possession of a corresponding decryption or encryption key, respectively; a common example is a "public key cryptographic system," in which possession of the encryption key does not make it practically feasible to deduce the decryption key, so that the encryption key may safely be made available to the public. An example of a public key cryptographic system is RSA, in which an encryption key involves the use of numbers that are products of very large prime numbers, but a decryption key involves the use of those very large prime numbers, such that deducing the decryption key from the encryption key requires the practically infeasible task of computing the prime factors of a number which is the product of two very large prime numbers. Another example is elliptic curve

cryptography, which relies on the fact that given two points P and Q on an elliptic curve over a finite field, and a definition for addition where A+B=−R, the point where a line connecting point A and point B intersects the elliptic curve, where "**0**," the identity, is a point at infinity in a projective plane containing the elliptic curve, finding a number k such that adding P to itself k times results in Q is computationally impractical, given correctly selected elliptic curve, finite field, and P and Q. A further example of asymmetrical cryptography may include lattice-based cryptography, which relies on the fact that various properties of sets of integer combination of basis vectors are hard to compute, such as finding the one combination of basis vectors that results in the smallest Euclidean distance. Embodiments of cryptography, whether symmetrical or asymmetrical, may include quantum-secure cryptography, defined for the purposes of this disclosure as cryptography that remains secure against adversaries possessing quantum computers; some forms of lattice-based cryptography, for instance, may be quantum-secure.

In some embodiments, systems and methods described herein produce cryptographic hashes, also referred to by the equivalent shorthand term "hashes." A cryptographic hash, as used herein, is a mathematical representation of a lot of data, such as files or blocks in a block chain as described in further detail below; the mathematical representation is produced by a lossy "one-way" algorithm known as a "hashing algorithm." Hashing algorithm may be a repeatable process; that is, identical lots of data may produce identical hashes each time they are subjected to a particular hashing algorithm. Because hashing algorithm is a one-way function, it may be impossible to reconstruct a lot of data from a hash produced from the lot of data using the hashing algorithm. In the case of some hashing algorithms, reconstructing the full lot of data from the corresponding hash using a partial set of data from the full lot of data may be possible only by repeatedly guessing at the remaining data and repeating the hashing algorithm; it is thus computationally difficult if not infeasible for a single computer to produce the lot of data, as the statistical likelihood of correctly guessing the missing data may be extremely low. However, the statistical likelihood of a computer of a set of computers simultaneously attempting to guess the missing data within a useful timeframe may be higher, permitting mining protocols as described in further detail below.

In an embodiment, hashing algorithm may demonstrate an "avalanche effect," whereby even extremely small changes to lot of data produce drastically different hashes. This may thwart attempts to avoid the computational work necessary to recreate a hash by simply inserting a fraudulent datum in data lot, enabling the use of hashing algorithms for "tamper-proofing" data such as data contained in an immutable ledger as described in further detail below. This avalanche or "cascade" effect may be evinced by various hashing processes; persons skilled in the art, upon reading the entirety of this disclosure, will be aware of various suitable hashing algorithms for purposes described herein. Verification of a hash corresponding to a lot of data may be performed by running the lot of data through a hashing algorithm used to produce the hash. Such verification may be computationally expensive, albeit feasible, potentially adding up to significant processing delays where repeated hashing, or hashing of large quantities of data, is required, for instance as described in further detail below. Examples of hashing programs include, without limitation, SHA256, a NIST standard; further current and past hashing algorithms include Winternitz hashing algorithms, various generations of Secure Hash Algorithm (including "SHA-1," "SHA-2," and "SHA-3"), "Message Digest" family hashes such as "MD4," "MD5," "MD6," and "RIPEMD," Keccak, "BLAKE" hashes and progeny (e.g., "BLAKE2," "BLAKE-256," "BLAKE-512," and the like), Message Authentication Code ("MAC")-family hash functions such as PMAC, OMAC, VMAC, HMAC, and UMAC, Poly1305-AES, Elliptic Curve Only Hash ("ECOH") and similar hash functions, Fast-Syndrome-based (FSB) hash functions, GOST hash functions, the Grøstl hash function, the HAS-160 hash function, the JH hash function, the RadioGatun hash function, the Skein hash function, the Streebog hash function, the SWIFFT hash function, the Tiger hash function, the Whirlpool hash function, or any hash function that satisfies, at the time of implementation, the requirements that a cryptographic hash be deterministic, infeasible to reverse-hash, infeasible to find collisions, and have the property that small changes to an original message to be hashed will change the resulting hash so extensively that the original hash and the new hash appear uncorrelated to each other. A degree of security of a hash function in practice may depend both on the hash function itself and on characteristics of the message and/or digest used in the hash function. For example, where a message is random, for a hash function that fulfills collision-resistance requirements, a brute-force or "birthday attack" may to detect collision may be on the order of O(2n/2) for n output bits; thus, it may take on the order of 2256 operations to locate a collision in a 512 bit output "Dictionary" attacks on hashes likely to have been generated from a non-random original text can have a lower computational complexity, because the space of entries they are guessing is far smaller than the space containing all random permutations of bits. However, the space of possible messages may be augmented by increasing the length or potential length of a possible message, or by implementing a protocol whereby one or more randomly selected strings or sets of data are added to the message, rendering a dictionary attack significantly less effective.

In some embodiments, apparatuses and methods described herein may generate, evaluate, and/or utilize digital signatures. A "digital signature," as used herein, includes a secure proof of possession of a secret by a signing device, as performed on provided element of data, known as a "message." A message may include an encrypted mathematical representation of a file or other set of data using the private key of a public key cryptographic system. Secure proof may include any form of secure proof as described above, including without limitation encryption using a private key of a public key cryptographic system as described above. Signature may be verified using a verification datum suitable for verification of a secure proof; for instance, where secure proof is enacted by encrypting message using a private key of a public key cryptographic system, verification may include decrypting the encrypted message using the corresponding public key and comparing the decrypted representation to a purported match that was not encrypted; if the signature protocol is well-designed and implemented correctly, this means the ability to create the digital signature is equivalent to possession of the private decryption key and/or device-specific secret. Likewise, if a message making up a mathematical representation of file is well-designed and implemented correctly, any alteration of the file may result in a mismatch with the digital signature; the mathematical representation may be produced using an alteration-sensitive, reliably reproducible algorithm, such as a hashing algorithm as described above. A mathematical representation to which the signature may be compared may be

included with signature, for verification purposes; in other embodiments, the algorithm used to produce the mathematical representation may be publicly available, permitting the easy reproduction of the mathematical representation corresponding to any file.

In some embodiments, digital signatures may be combined with or incorporated in digital certificates. In one embodiment, a digital certificate is a file that conveys information and links the conveyed information to a "certificate authority" that is the issuer of a public key in a public key cryptographic system. Certificate authority in some embodiments contains data conveying the certificate authority's authorization for the recipient to perform a task. The authorization may be the authorization to access a given datum. The authorization may be the authorization to access a given process. In some embodiments, the certificate may identify the certificate authority. The digital certificate may include a digital signature. A third party such as a certificate authority (CA) is available to verify that the possessor of the private key is a particular entity; thus, if the certificate authority may be trusted, and the private key has not been stolen, the ability of an entity to produce a digital signature confirms the identity of the entity and links the file to the entity in a verifiable way. Digital signature may be incorporated in a digital certificate, which is a document authenticating the entity possessing the private key by authority of the issuing certificate authority and signed with a digital signature created with that private key and a mathematical representation of the remainder of the certificate. In other embodiments, digital signature is verified by comparing the digital signature to one known to have been created by the entity that purportedly signed the digital signature; for instance, if the public key that decrypts the known signature also decrypts the digital signature, the digital signature may be considered verified. Digital signature may also be used to verify that the file has not been altered since the formation of the digital signature.

Referring now to FIG. **1**, an exemplary embodiment of an apparatus **100** for using machine learning to verify lost user data in a resource data storage system is illustrated. Apparatus **100** includes a processor **104** and a memory **108** communicatively connected to processor **104**, wherein memory **108** contains instructions configuring processor **104** to carry out the verifying process. Processor **104** and memory **108** is contained in a computing device **112**. As used in this disclosure, "communicatively connected" means connected by way of a connection, attachment, or linkage between two or more relata which allows for reception and/or transmittance of information therebetween. For example, and without limitation, this connection may be wired or wireless, direct, or indirect, and between two or more components, circuits, devices, systems, and the like, which allows for reception and/or transmittance of data and/or signal(s) therebetween. Data and/or signals therebetween may include, without limitation, electrical, electromagnetic, magnetic, video, audio, radio, and microwave data and/or signals, combinations thereof, and the like, among others. A communicative connection may be achieved, for example and without limitation, through wired or wireless electronic, digital, or analog, communication, either directly or by way of one or more intervening devices or components. Further, communicative connection may include electrically coupling or connecting at least an output of one device, component, or circuit to at least an input of another device, component, or circuit. For example, and without limitation, via a bus or other facility for intercommunication between elements of a computing device. Communicative connecting may also include indirect connections via, for example and without limitation, wireless connection, radio communication, low power wide area network, optical communication, magnetic, capacitive, or optical coupling, and the like. In some instances, the terminology "communicatively coupled" may be used in place of communicatively connected in this disclosure. A computing device **112** may include any computing device as described in this disclosure, including without limitation a microcontroller, microprocessor, digital signal processor (DSP) and/or system on a chip (SoC) as described in this disclosure. Computing device **112** may include, be included in, and/or communicate with a mobile device such as a mobile telephone or smartphone. Computing device **112** may include a single computing device operating independently, or may include two or more computing device operating in concert, in parallel, sequentially or the like; two or more computing devices may be included together in a single computing device or in two or more computing devices. Computing device **112** may interface or communicate with one or more additional devices as described below in further detail via a network interface device. Network interface device may be utilized for connecting computing device **112** to one or more of a variety of networks, and one or more devices. Examples of a network interface device include, but are not limited to, a network interface card (e.g., a mobile network interface card, a LAN card), a modem, and any combination thereof. Examples of a network include, but are not limited to, a wide area network (e.g., the Internet, an enterprise network), a local area network (e.g., a network associated with an office, a building, a campus or other relatively small geographic space), a telephone network, a data network associated with a telephone/voice provider (e.g., a mobile communications provider data and/or voice network), a direct connection between two computing devices, and any combinations thereof. A network may employ a wired and/or a wireless mode of communication. In general, any network topology may be used. Information (e.g., data, software etc.) may be communicated to and/or from a computer and/or a computing device. Computing device **112** may include but is not limited to, for example, a computing device or cluster of computing devices in a first location and a second computing device or cluster of computing devices in a second location. Computing device **112** may include one or more computing devices dedicated to data storage, security, distribution of traffic for load balancing, and the like. Computing device **112** may distribute one or more computing tasks as described below across a plurality of computing devices of computing device, which may operate in parallel, in series, redundantly, or in any other manner used for distribution of tasks or memory between computing devices. Computing device **112** may be implemented using a "shared nothing" architecture in which data is cached at the worker, in an embodiment, this may enable scalability of apparatus **100** and/or computing device **112**.

With continued reference to FIG. **1**, processor **104** and/or computing device **112** may be designed and/or configured by memory **108** to perform any method, method step, or sequence of method steps in any embodiment described in this disclosure, in any order and with any degree of repetition. For instance, processor **104** and/or computing device **112** may be configured to perform a single step or sequence repeatedly until a desired or commanded outcome is achieved; repetition of a step or a sequence of steps may be performed iteratively and/or recursively using outputs of previous repetitions as inputs to subsequent repetitions, aggregating inputs and/or outputs of repetitions to produce

an aggregate result, reduction or decrement of one or more variables such as global variables, and/or division of a larger processing task into a set of iteratively addressed smaller processing tasks. Processor **104** and/or computing device **112** may perform any step or sequence of steps as described in this disclosure in parallel, such as simultaneously and/or substantially simultaneously performing a step two or more times using two or more parallel threads, processor cores, or the like; division of tasks between parallel threads and/or processes may be performed according to any protocol suitable for division of tasks between iterations. Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various ways in which steps, sequences of steps, processing tasks, and/or data may be subdivided, shared, or otherwise dealt with using iteration, recursion, and/or parallel processing.

Processor **104** and/or computing device **112** may perform determinations, classification, and/or analysis steps, methods, processes, or the like as described in this disclosure using machine learning processes **116**. A "machine learning process," as used in this disclosure, is a process that automatedly uses a body of data known as "training data" and/or a "training set" (described further below) to generate an algorithm that will be performed by a computing device/module to produce outputs given data provided as inputs; this is in contrast to a non-machine learning software program where the commands to be executed are determined in advance by a user and written in a programming language. Machine-learning process **116** may utilize supervised, unsupervised, lazy-learning processes and/or neural networks, described further below.

Still referring to FIG. **1**, processor **104** and/or computing device **112** is configured to identify a plurality of lost user data **120** stored in a resource data storage system potentially associated with a particular user of a plurality of users. As used in this disclosure, "lost user data" is a data record, as defined below, that is not properly associated with a user participating in the recourse data storage system. Lost user data **120** may be a data record that could not be verified and properly added to the resource data storage system at the time of its creation by a user. A "user" is a jobseeker or applicant, an administrator (e.g. apparatus administrator of the immutable sequential listing **128** network), a professional reference, and the like. "Potentially associated", with in this disclosure, means lost user data **120** in the resource data storage system that has not been verified/digitally signed but based on historical data pertaining to the resource storage system **124** may match with a particular user of the plurality of users. Matching may be the classification of lost user data to a particular user by processor **104** and/or computing device **112** using a machine learning classification algorithm, as described further below. Matching may occur prior to verification of lost user data **120**, thus making lost user data **120** only potentially associated instead of fully associated with a particular user. Historical data may be used to train the classification algorithm to output matches between lost user data **120** and a user. The historical data may include a plurality of similar verified user data associated with a plurality of users. For example, lost user data **120** may be a cover letter, wherein the historical data of resource storage system **124** contains the IP address of the device used to upload the cover letter onto immutable sequential listing **128** and the plurality of IP addresses associated with the plurality of users that are similar to the cover letter address.

Still referring to FIG. **1**, as used herein, a "resource data storage system" is a database stored on processor **104** and/or computing device **112** or stored remotely that contains information on a job candidate. In an embodiment, information may include user-specific historical data, other job information, education information, or the like. Resource data storage system is stored on immutable sequential listing **128**. An "immutable sequential listing," as used in this disclosure, is a data structure that places data entries in a fixed sequential arrangement, such as a temporal sequence of entries and/or blocks thereof, where the sequential arrangement, once established, cannot be altered or reordered. As used in this disclosure, a "data record" is a record containing one or more data elements within immutable sequential listing **128**, where "within" indicates recordation within immutable sequential listing **128**, including entry of a representation using a hash, cryptographic accumulator, or other element of data representing the data record, for instance and without limitation as described in further detail below. The resource data storage system may contain entries representing data of at least a job candidate. For example, a job board website may store candidate information such as but not limited to age, name, gender, title at past or current position, field of work, experience, events, career milestones, educational attainments, etc. Entries of data may also include records of transactions, such as Bitcoin transactions. Additionally, entries of data may include files, such as JPEGs, documents, spreadsheets, videos, pictures, etc. Blocks of immutable sequential listing **128** may be hashed and encoded into a Merkle tree. In an embodiment, each block includes the cryptographic hash of the prior block, linking the blocks and creating a chain. The top of the Merkle tree may include a Merkle root that may include a cryptographic accumulator. Immutable sequential listing **128** may include a cryptographic accumulator. A "cryptographic accumulator," as used in this disclosure, is a data structure created by relating a commitment, which may be smaller amount of data that may be referred to as an "accumulator" and/or "root," to a set of elements, such as lots of data and/or collection of data, together with short membership and/or non-membership proofs for any element in the set. In an embodiment, these proofs may be publicly verifiable against the commitment. An accumulator may be said to be "dynamic" if the commitment and membership proofs can be updated efficiently as elements are added or removed from the set, at unit cost independent of the number of accumulated elements; an accumulator for which this is not the case may be referred to as "static." A membership proof may be referred to as a as a "witness" whereby an element existing in the larger amount of data can be shown to be included in the root, while an element not existing in the larger amount of data can be shown not to be included in the root, where "inclusion" indicates that the included element was a part of the process of generating the root, and therefore was included in the original larger data set.

In some cases, a data record may include user's resume or profile, which may be in video, visual, audio and/or text format, a cover letter and other information relating to the applicant's education, experience (professional, work, personal), qualifications, interests, and the like, among others. Sometimes separate blocks can be produced concurrently, creating a temporary fork in immutable sequential listing **128**. In addition to a secure hash-based history, immutable sequential listing **128** may have a specified algorithm for scoring different versions of the history so that one with a higher score can be selected over others. Blocks not selected for inclusion in the chain are called orphan blocks. In some embodiments, the plurality of lost user data **120** may include

a plurality of orphan blocks in a cryptographic hash chain existing outside of a main hash-chain on immutable sequential listing **128**.

With continued reference to FIG. **1**, in some embodiments, processor **104** and/or computing device **112** may be configured to run as a full node to monitor the creation of blocks to identify lost user data **120**. A full node is a device that contains a full copy of the transaction history of immutable sequential listing **128**. Nodes can be any kind of device (mostly computers, laptops or even bigger servers). Nodes may form the infrastructure of immutable sequential listing **128**. All nodes on immutable sequential listing **128** may be connected to each other through a network and may constantly exchange the latest immutable sequential listing **128** data with each other, so all nodes stay up to date.

Still referring to FIG. **1**, identifying a plurality of lost user data **120** includes dividing the plurality of lost user data **120** into a plurality of user identifier subsets **144** that are identified by at least a public key **132**, wherein each of the plurality of user identifier subsets **144** is matched with the particular user of the plurality of users. As used in this disclosure, a "user identifier subset" is categorized data entries, such as sections of user data pertaining to work, experiences, hobbies, address, names/titles, and the like. A user may have multiple user identifier subsets **144** which may be implemented, for instance, as immutable sequential listing **128** addresses, public keys **132** of public-key cryptographic systems, or the like. For example, a user may have a subset related to internship experience and another for published research papers, both of which may be associated with public key **132** the user may access using a private key. In some embodiments, dividing lost user data **120** into user identifier subsets **144** may include using machine learning process **116** to output data bins of matched lost user data **120** to a particular user. Machine learning process **116** may include using classifier **136**. A "classifier," as used in this disclosure is a machine-learning model, such as a mathematical model, neural net, or program generated by a machine learning algorithm known as a "classification algorithm," as described in further detail below, that sorts inputs into categories or bins of data, outputting the categories or bins of data and/or labels associated therewith. Classifier **136** may be configured to output at least a datum that labels or otherwise identifies a set of data that are clustered together, found to be close under a distance metric as described below, or the like. Processor **104** and/or computing device **112** and/or another device may generate a classifier using a classification algorithm, defined as a processes whereby a processor **104** and/or computing device **112** derives a classifier from training data. In an embodiment, a user identifier subset classifier may take a plurality of lost user data **120** as an algorithm input, and output data bins containing matches of subset to a plurality of user identifiers. The training data may include historical data such as the transaction history of immutable sequential listing **128**, IP Address of users and blocks at time of creation, publics keys and/ or immutable sequential listing **128** address associated with a plauitry of user, and the like.

Classification may be performed using, without limitation, linear classifiers such as without limitation logistic regression and/or naive Bayes classifiers, nearest neighbor classifiers such as k-nearest neighbors classifiers, support vector machines, least squares support vector machines, fisher's linear discriminant, quadratic classifiers, decision trees, boosted trees, random forest classifiers, learning vector quantization, and/or neural network-based classifiers.

Still referring to FIG. **1**, processor **104** and/or computing device **112** may be configured to generate a classifier using a Naive Bayes classification algorithm. Naive Bayes classification algorithm generates classifiers by assigning class labels to problem instances, represented as vectors of element values. Class labels are drawn from a finite set. Naive Bayes classification algorithm may include generating a family of algorithms that assume that the value of a particular element is independent of the value of any other element, given a class variable. Naive Bayes classification algorithm may be based on Bayes Theorem expressed as $P(A/B)=P(B/A)\,P(A)\div P(B)$, where $P(A/B)$ is the probability of hypothesis A given data B also known as posterior probability; $P(B/A)$ is the probability of data B given that the hypothesis A was true; $P(A)$ is the probability of hypothesis A being true regardless of data also known as prior probability of A; and $P(B)$ is the probability of the data regardless of the hypothesis. A naive Bayes algorithm may be generated by first transforming training data into a frequency table. Processor **104** and/or computing device **112** may then calculate a likelihood table by calculating probabilities of different data entries and classification labels. Processor **104** and/or computing device **112** may utilize a naive Bayes equation to calculate a posterior probability for each class. A class containing the highest posterior probability is the outcome of prediction. Naive Bayes classification algorithm may include a gaussian model that follows a normal distribution. Naive Bayes classification algorithm may include a multinomial model that is used for discrete counts. Naive Bayes classification algorithm may include a Bernoulli model that may be utilized when vectors are binary.

With continued reference to FIG. **1**, processor **104** and/or computing device **112** may be configured to generate a classifier using a K-nearest neighbors (KNN) algorithm. A "K-nearest neighbors algorithm" as used in this disclosure, includes a classification method that utilizes feature similarity to analyze how closely out-of-sample- features resemble training data to classify input data to one or more clusters and/or categories of features as represented in training data; this may be performed by representing both training data and input data in vector forms, and using one or more measures of vector similarity to identify classifications within training data, and to determine a classification of input data. K-nearest neighbors algorithm may include specifying a K-value, or a number directing the classifier to select the k most similar entries training data to a given sample, determining the most common classifier of the entries in the database, and classifying the known sample; this may be performed recursively and/or iteratively to generate a classifier that may be used to classify input data as further samples. For instance, an initial set of samples may be performed to cover an initial heuristic and/or "first guess" at an output and/or relationship, which may be seeded, without limitation, using expert input received according to any process as described herein. As a non-limiting example, an initial heuristic may include a ranking of associations between inputs and elements of training data. Heuristic may include selecting some number of highest-ranking associations and/or training data elements.

With continued reference to FIG. **1**, generating k-nearest neighbors algorithm may generate a first vector output containing a data entry cluster, generating a second vector output containing an input data, and calculate the distance between the first vector output and the second vector output using any suitable norm such as cosine similarity, Euclidean distance measurement, or the like. Each vector output may be represented, without limitation, as an n-tuple of values,

where n is at least two values. Each value of n-tuple of values may represent a measurement or other quantitative value associated with a given category of data, or attribute, examples of which are provided in further detail below; a vector may be represented, without limitation, in n-dimensional space using an axis per category of value represented in n-tuple of values, such that a vector has a geometric direction characterizing the relative quantities of attributes in the n-tuple as compared to each other. Two vectors may be considered equivalent where their directions, and/or the relative quantities of values within each vector as compared to each other, are the same; thus, as a non-limiting example, a vector represented as [5, 10, 15] may be treated as equivalent, for purposes of this disclosure, as a vector represented as [1, 2, 3]. Vectors may be more similar where their directions are more similar, and more different where their directions are more divergent; however, vector similarity may alternatively or additionally be determined using averages of similarities between like attributes, or any other measure of similarity suitable for any n-tuple of values, or aggregation of numerical similarity measures for the purposes of loss functions as described in further detail below. Any vectors as described herein may be scaled, such that each vector represents each attribute along an equivalent scale of values. Each vector may be "normalized," or divided by a "length" attribute, such as a length attribute l as derived using a Pythagorean norm: l =

$$\sqrt{\sum_{i=0}^{n} a_i^2},$$

wnere a_i is attribute number i of the vector. Scaling and/or normalization may function to make vector comparison independent of absolute quantities of attributes, while preserving any dependency on similarity of attributes; this may, for instance, be advantageous where cases represented in training data are represented by different quantities of samples, which may result in proportionally equivalent vectors with divergent values.

Still referring to FIG. **1**, Processor **104** and/or computing device **112** is configured to verify, using a recovery program **140**, a plurality of lost user data **120** potentially associated with a particular user. As used in this disclosure, a "recovery program" is a method of salvaging deleted, inaccessible, lost, corrupted, damaged, or formatted data from secondary storage, removable media, or files, when the data stored in them cannot be accessed in a usual way. In some embodiments, recovery program **140** may include sending a user an electronic request, such as an automated email, for verification using public-key cryptography, as disclosed above, wherein user identifier subsets **144** associated with public key **132** (which may be known to others) may be recovered by a user digitally signing with a private key (which may not be known by anyone except the user). A public key, as discussed above, is an algorithm (such as RSA) that encrypts information but does not decrypt the same information. In an embodiment, the user may get a verification request on another processor **104** and/or computing device **112** known to be owned by the user. For example, a user may use a private key to decrypt public key **132** of a block or hash containing the lost user date. The user then may be allowed to associate lost user data **120** with themselves with a digital signature.

In some embodiments, verification may include proof of possession, as defined above, wherein a digital signature may be signed with a private key such that decrypting the block with public key **132** proves possession. Proof of possession may include a zero-knowledge proof, which may provide an output demonstrating possession of a secret while revealing none of the secret to a recipient of the output; zero-knowledge proof may be information-theoretically secure, meaning that an entity with infinite computing power would be unable to determine secret from output. Alternatively, zero-knowledge proof may be computationally secure, meaning that determination of secret from output is computationally infeasible, for instance to the same extent that determination of a private key from public key **132** in a public key cryptographic system is computationally infeasible. Zero-knowledge proof algorithms may generally include a set of two algorithms, a prover algorithm, or "P," which is used to prove computational integrity and/or possession of a secret, and a verifier algorithm, or "V" whereby a party may check the validity of P. Zero-knowledge proof may include an interactive zero-knowledge proof, wherein a party verifying the proof must directly interact with the proving party; for instance, the verifying and proving parties may be required to be online, or connected to the same network as each other, at the same time. Interactive zero-knowledge proof may include a "proof of knowledge" proof, such as a Schnorr algorithm for proof on knowledge of a discrete logarithm. in a Schnorr algorithm, a prover commits to a randomness r, generates a message based on r, and generates a message adding r to a challenge c multiplied by a discrete logarithm that the prover is able to calculate; verification is performed by the verifier who produced c by exponentiation, thus checking the validity of the discrete logarithm. Interactive zero-knowledge proofs may alternatively or additionally include sigma protocols. Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various alternative interactive zero-knowledge proofs that may be implemented consistently with this disclosure.

Alternatively, zero-knowledge proof may include a non-interactive zero-knowledge, proof, or a proof wherein neither party to the proof interacts with the other party to the proof; for instance, each of a party receiving the proof and a party providing the proof may receive a reference datum which the party providing the proof may modify or otherwise use to perform the proof. As a non-limiting example, zero-knowledge proof may include a succinct non-interactive arguments of knowledge (ZK-SNARKS) proof, wherein a "trusted setup" process creates proof and verification keys using secret (and subsequently discarded) information encoded using a public key cryptographic system, a prover runs a proving algorithm using the proving key and secret information available to the prover, and a verifier checks the proof using the verification key; public key cryptographic system may include RSA, elliptic curve cryptography, E1Gamal, or any other suitable public key cryptographic system. Generation of trusted setup may be performed using a secure multiparty computation so that no one party has control of the totality of the secret information used in the trusted setup; as a result, if any one party generating the trusted setup is trustworthy, the secret information may be unrecoverable by malicious parties. As another non-limiting example, non-interactive zero-knowledge proof may include a Succinct Transparent Arguments of Knowledge (ZK-STARKS) zero-knowledge proof. In an embodiment, a ZK-STARKS proof includes a Merkle root of a Merkle tree representing evaluation of a secret computation at some

number of points, which may be 1 billion points, plus Merkle branches representing evaluations at a set of randomly selected points of the number of points; verification may include determining that Merkle branches provided match the Merkle root, and that point verifications at those branches represent valid values, where validity is shown by demonstrating that all values belong to the same polynomial created by transforming the secret computation. In an embodiment, ZK-STARKS does not require a trusted setup.

Zero-knowledge proof may include any other suitable zero-knowledge proof. Zero-knowledge proof may include, without limitation bulletproofs. Zero-knowledge proof may include a homomorphic public-key cryptography (hPKC)-based proof. Zero-knowledge proof may include a discrete logarithmic problem (DLP) proof. Zero-knowledge proof may include a secure multi-party computation (MPC) proof. Zero-knowledge proof may include, without limitation, an incrementally verifiable computation (IVC). Zero-knowledge proof may include an interactive oracle proof (IOP). Zero-knowledge proof may include a proof based on the probabilistically checkable proof (PCP) theorem, including a linear PCP (LPCP) proof. Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various forms of zero-knowledge proofs that may be used, singly or in combination, consistently with this disclosure.

In an embodiment, proof of possession is implemented using a challenge-response protocol. In an embodiment, this may function as a one-time pad implementation; for instance, a manufacturer or other trusted party may record a series of outputs ("responses") produced by a device possessing secret information, given a series of corresponding inputs ("challenges"), and store them securely. In an embodiment, a challenge-response protocol may be combined with key generation. A single key may be used in one or more digital signatures as described in further detail below, such as signatures used to receive and/or transfer possession of crypto-currency assets; the key may be discarded for future use after a set period of time. In an embodiment, varied inputs include variations in local physical parameters, such as fluctuations in local electromagnetic fields, radiation, temperature, and the like, such that an almost limitless variety of private keys may be so generated. Proof of possession may include encryption of a challenge to produce the response, indicating possession of a secret key. Encryption may be performed using a private key of a public key cryptographic system, or using a private key of a symmetric cryptographic system; for instance, trusted party may verify response by decrypting an encryption of challenge or of another datum using either a symmetric or public-key cryptographic system, verifying that a stored key matches the key used for encryption as a function of at least a device-specific secret. Keys may be generated by random variation in selection of prime numbers, for instance for the purposes of a cryptographic system such as RSA that relies prime factoring difficulty. Keys may be generated by randomized selection of parameters for a seed in a cryptographic system, such as elliptic curve cryptography, which is generated from a seed. Keys may be used to generate exponents for a cryptographic system such as Diffie-Helman or ElGamal that are based on the discrete logarithm problem.

Still referring to FIG. **1**, in some embodiments, verifying a plurality of lost user data **120** may include using a cryptographic commitment. A "commitment", as used herein, is a cryptographic algorithm that allows the user to commit to a certain value without revealing it. In this case, a user's identity may be verified by opening the commitment. For example, a user may be requested by processor **104** and/or computing device **112** to enter their personal identification number (PIN) as a commitment. The commitment may be hashed and compared to a hash of the user-specific secret. If the two hashes are identical, user identity may be verified. As used in this disclosure, user-specific secret (also referred to as "secret") includes any data that is only known by or only possessed by the user. For example, a secret may include a password, a personal identification number, a mnemonic device, etc. Cryptographic commitment may include a Pedersen commitment. This may be used to verify user identity to prove possession of lost user data **120** later on when the commitment is opened. A "Pedersen commitment", as used herein is a specific type of commitment that uses a secret message with at least two elements, a random secret, and a commitment algorithm that produces a commitment as a function of the secret message and a random secret. A receiver/verifier is given the commitment, secret message, and random secret and can verify the commitment by putting the secret message and random secret back into the commitment algorithm. A cryptographic commitment may additionally or alternatively include a cryptographic hash of the user-specific secret, and/or a cryptographic accumulator such as a Merkle tree of the user-specific secret. In an example where a user password is the user-specific secret, a hash of the commitment may be compared to the hash of the actual user password to verify user identity. Additionally or alternative, a commitment may use a personal identification number, mnemonic device, biometric key/datum, and the like. In another embodiment, a commitment may be verified by determining that a cryptographic accumulator contains the user-specific secret. Additional information on verifying user data is illustrated in U.S. patent application Ser. No. 17/667,711 entitled "APPARATUS AND METHODS FOR VALIDATING USER DATA" filed on Feb. 9, 2021, which is incorporated by reference herein entirely.

Still referring to FIG. **1**, in some embodiments, verifying a plurality of lost user data **120** may include using a classifier to match the plurality of lost user data **120** to a plurality of user based on a generated confidence level. As used in this disclosure, a "confidence level", is an element of data expressing a degree to which the safety, security, or authenticity of a process, device, or datum may be relied upon. As used herein, a confidence level may include a numerical score; numerical score may be a score on a scale having one extremum representing a maximal degree of reliability, and a second extremum representing a minimum degree of liability. As a non-limiting example, extremum representing maximal degree of reliability may be a maximal number of an ordered set of numbers such as an open or closed set on the real number line, a sequential listing of integers or natural numbers, or the like; persons skilled in the art will be aware that selection of a numerical extremum to represent a higher level of confidence or reliability, albeit intuitively pleasing, is not mathematically necessary, and any suitable mapping of level of confidence or reliability to numerical objects or ranges may feasibly be substituted. As a further non-limiting example, numerical score may include, or be mappable to, a probability score, such as a percentage probability or a 0-1 probability level. Confidence level may include further information or indications, such as without limitation flags denoting untrustworthy, suspect, or hostile elements; for instance a flag may indicate that a particular device, program, process, or element of data appears to be compromised and/or has been involved in fraudulent or otherwise hostile or disruptive engagement with apparatus **100**. In some embodiments, a plurality of lost user data **120**

may be used as an input to the algorithm and the output may be a data bins the plurality of user categorized by a confidence level. Training data may include historical data such as the transaction history of immutable sequential listing **128**, IP Address of users and blocks at time of creation, publics keys and/or immutable sequential listing **128** address associated with a plurality of users, output data bins of classifier **136**, digital certificates, secure proofs, digital signatures of the plurality of users, and the like. A desired confidence level to be used as a threshold may be computed in turn by reference to a user input indicating a desired confidence level, a minimal confidence level set by processor **104** and/or computing device **112**, for instance to ensure some degree of overall network integrity, a calculation based on a value of a transaction recorded in at least a digitally signed assertion, or the like. For example, lost user data **120** may be assigned a confidence level that exceeds the threshold requirement based on historical data pertaining to how recently a user performed a digitally assigned assertion in an verified instance on immutable sequential listing **128**.

Still referring to FIG.**1**, the classifier may operate as a k-means clustering algorithm receiving unclassified semantic elements and outputs a definite number of classified data entry clusters wherein the data entry clusters each contain cluster data entries. K-means algorithm may select a specific number of groups or clusters to output, identified by a variable "k." Generating a k- means clustering algorithm includes assigning inputs containing unclassified data to a "k-group " or "k-cluster" based on feature similarity. Centroids of k-groups or k-clusters may be utilized to generate classified data entry cluster. K-means clustering algorithm may select and/or be provided "k" variable by calculating k-means clustering algorithm for a range of k values and comparing results. K-means clustering algorithm may compare results across different values of k as the mean distance between cluster data entries and cluster centroid. K-means clustering algorithm may calculate mean distance to a centroid as a function of k value, and the location of where the rate of decrease starts to sharply shift, this may be utilized to select a k value. Centroids of k-groups or k-cluster include a collection of feature values which are utilized to classify data entry clusters containing cluster data entries. K-means clustering algorithm may act to identify clusters of closely related semantic element data, which may be provided with similar semantic element data; this may, for instance, generate an initial set of similar semantic element data from an initial set of semantic element data of a large number of users, and may also, upon subsequent iterations, identify new clusters to be provided new similar semantic element data, to which additional semantic element data may be classified, or to which previously used semantic element data may be reclassified.

With continued reference to FIG. **1**, generating a k-means clustering algorithm may include generating initial estimates for k centroids which may be randomly generated or randomly selected from unclassified data input. K centroids may be utilized to define one or more clusters. K-means clustering algorithm may assign unclassified data to one or more k-centroids based on the squared Euclidean distance by first performing a data assigned step of unclassified data. K-means clustering algorithm may assign unclassified data to its nearest centroid based on the collection of centroids c_i of centroids in set C. Unclassified data may be assigned to a cluster based on $\text{argmin}_{ci \ni C} \text{dist}(ci, x)^2$, where argmin includes argument of the minimum, ci includes a collection of centroids in a set C, and dist includes standard Euclidean distance. K-means clustering module may then recompute centroids by taking mean of all cluster data entries assigned to a centroid's cluster. This may be calculated based on $ci=1/|Si|\Sigma xi \ni Si^{xi}$. K-means clustering algorithm may continue to repeat these calculations until a stopping criterion has been satisfied such as when cluster data entries do not change clusters, the sum of the distances have been minimized, and/or some maximum number of iterations has been reached.

Still referring to FIG. **1**, k-means clustering algorithm may be configured to calculate a degree of similarity index value. A "degree of similarity index value" as used in this disclosure, includes a distance measurement indicating a measurement between each data entry cluster generated by k-means clustering algorithm and a selected semantic element set. Degree of similarity index value may indicate how close a particular combination of semantic element data, similar semantic element data and/or semantic data of opportunity listings is to being classified by k-means algorithm to a particular cluster. K-means clustering algorithm may evaluate the distances of the combination of similar semantic element data to the k-number of clusters output by k-means clustering algorithm. Short distances between a set of semantic element data and a cluster may indicate a higher degree of similarity between the set of semantic element data and a particular cluster. Longer distances between a set of semantic element data and a cluster may indicate a lower degree of similarity between a semantic element data set and a particular cluster.

With continued reference to FIG. **1**, k-means clustering algorithm selects a classified data entry cluster as a function of the degree of similarity index value. In an embodiment, k-means clustering algorithm may select a classified data entry cluster with the smallest degree of similarity index value indicating a high degree of similarity between a semantic element data set and the data entry cluster. Alternatively or additionally k-means clustering algorithm may select a plurality of clusters having low degree of similarity index values to semantic element data sets, indicative of greater degrees of similarity. Degree of similarity index values may be compared to a threshold number indicating a minimal degree of relatedness suitable for inclusion of a set of semantic element data in a cluster, where a degree of similarity indices falling under the threshold number may be included as indicative of high degrees of relatedness. Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various additional or alternative feature learning approaches that may be used consistently with this disclosure.

Still referring to FIG. **1**, in some embodiments, verifying a plurality of lost user data **120** may include using mathematical relationships between elements within lost user data **120**, such as user identifier subsets **144**, and previously verified user data elements of a potential user. In some embodiments, statistical relationships are determined through one or more machine learning processes; for instance, data describing the speed, authenticity, and anonymity of a plurality of past previously verified user data may be subjected to regression analysis, such as linear or polynomial regression, to determine one or more equations relating one parameter of such previously verified user data to one or more other such parameters. Similarly, a neural net may be provided with such a plurality of previously verified data. Machine-learning processes may be supervised and/or unsupervised; for instance, user identifier subsets **144** to compare may be preselected to ensure that machine-learning processes result in relationships between user identifier subsets **144** and previously verified user data parameters.

Mathematical relationships may demonstrate, e.g., that a certain number of nodes in at least a highly trusted node results in a 95% degree of confidence in authenticity, that a second, higher number of nodes results in a 98% degree of confidence, and the like.

Still referring to FIG. **1**, in some embodiments, verifying a plurality of lost user data **120** may include using fuzzy set comparison as described further below, wherein lost user data **120** may be inputted into a fuzzy set and compared against a fuzzy set related to a user containing previously verified user data elements.

Still referring to FIG.**1**, Processor **104** and/or computing device **112** is configured to link a plurality of verified lost user data **120** to a particular user on immutable sequential listing **128**. In some embodiments, linking may include cryptographically inserting into the main hash-chain on immutable sequential listing **128**, the plurality of lost user data **120** associated with the particular user. For example, a new block may be added to immutable sequential listing **128** containing the newly matched and verified user associated with the particular user, using any hashing process as described throughout this disclosure. In an embodiment, where validation is a condition to linkage, processor **104** and/or computing device **112** may only cryptographically insert into the main-hash-chain, the first data bin containing the plurality of verified lost user data **120** that was able to be validated by a classifier as described above. The second data bin containing the plurality of verified lost user data **120** that could not be validated, may be archived separate from the main hash-chain.

Referring now to FIG. **2**, an exemplary embodiment of an immutable sequential listing **200** is illustrated. Data elements are listing in immutable sequential listing **200**; data elements may include any form of data, including textual data, image data, encrypted data, cryptographically hashed data, and the like. Data elements may include, without limitation, one or more at least a digitally signed assertions. In one embodiment, a digitally signed assertion **204** is a collection of textual data signed using a secure proof as described in further detail below; secure proof may include, without limitation, a digital signature as described above. Collection of textual data may contain any textual data, including without limitation American Standard Code for Information Interchange (ASCII), Unicode, or similar computer-encoded textual data, any alphanumeric data, punctuation, diacritical mark, or any character or other marking used in any writing system to convey information, in any form, including any plaintext or cyphertext data; in an embodiment, collection of textual data may be encrypted, or may be a hash of other data, such as a root or node of a Merkle tree or hash tree, or a hash of any other information desired to be recorded in some fashion using a digitally signed assertion **204**. In an embodiment, collection of textual data states that the owner of a certain transferable item represented in a digitally signed assertion **204** register is transferring that item to the owner of an address. A digitally signed assertion **204** may be signed by a digital signature created using the private key associated with the owner's public key, as described above.

Still referring to FIG. **2**, a digitally signed assertion **204** may describe a transfer of virtual currency, such as cryptocurrency as described below. The virtual currency may be a digital currency. Item of value may be a transfer of trust, for instance represented by a statement vouching for the identity or trustworthiness of the first entity. Item of value may be an interest in a fungible negotiable financial instrument representing ownership in a public or private corporation, a creditor relationship with a governmental body or a corporation, rights to ownership represented by an option, derivative financial instrument, commodity, debt-backed security such as a bond or debenture or other security as described in further detail below. A resource may be a physical machine e.g. a ride share vehicle or any other asset. A digitally signed assertion **204** may describe the transfer of a physical good; for instance, a digitally signed assertion **204** may describe the sale of a product. In some embodiments, a transfer nominally of one item may be used to represent a transfer of another item; for instance, a transfer of virtual currency may be interpreted as representing a transfer of an access right; conversely, where the item nominally transferred is something other than virtual currency, the transfer itself may still be treated as a transfer of virtual currency, having value that depends on many potential factors including the value of the item nominally transferred and the monetary value attendant to having the output of the transfer moved into a particular user's control. The item of value may be associated with a digitally signed assertion **204** by means of an exterior protocol, such as the COLORED COINS created according to protocols developed by The Colored Coins Foundation, the MASTERCOIN protocol developed by the Mastercoin Foundation, or the ETHEREUM platform offered by the Stiftung Ethereum Foundation of Baar, Switzerland, the Thunder protocol developed by Thunder Consensus, or any other protocol.

Still referring to FIG. **2**, in one embodiment, an address is a textual datum identifying the recipient of virtual currency or another item of value in a digitally signed assertion **204**. In some embodiments, address is linked to a public key, the corresponding private key of which is owned by the recipient of a digitally signed assertion **204**. For instance, address may be the public key. Address may be a representation, such as a hash, of the public key. Address may be linked to the public key in memory of a processor **104**, for instance via a "wallet shortener" protocol. Where address is linked to a public key, a transferee in a digitally signed assertion **204** may record a subsequent a digitally signed assertion **204** transferring some or all of the value transferred in the first a digitally signed assertion **204** to a new address in the same manner. A digitally signed assertion **204** may contain textual information that is not a transfer of some item of value in addition to, or as an alternative to, such a transfer. For instance, as described in further detail below, a digitally signed assertion **204** may indicate a confidence level associated with a distributed storage node as described in further detail below.

In an embodiment, and still referring to FIG. **2** immutable sequential listing **200** records a series of at least a posted content in a way that preserves the order in which the at least a posted content took place. Temporally sequential listing may be accessible at any of various security settings; for instance, and without limitation, temporally sequential listing may be readable and modifiable publicly, may be publicly readable but writable only by entities and/or devices having access privileges established by password protection, confidence level, or any device authentication procedure or facilities described herein, or may be readable and/or writable only by entities and/or devices having such access privileges. Access privileges may exist in more than one level, including, without limitation, a first access level or community of permitted entities and/or devices having ability to read, and a second access level or community of permitted entities and/or devices having ability to write; first and second community may be overlapping or non-overlapping. In an embodiment, posted content and/or immutable

sequential listing **200** may be stored as one or more zero knowledge sets (ZKS), Private Information Retrieval (PIR) structure, or any other structure that allows checking of membership in a set by querying with specific properties. Such database may incorporate protective measures to ensure that malicious actors may not query the database repeatedly in an effort to narrow the members of a set to reveal uniquely identifying information of a given posted content.

Still referring to FIG. **2**, immutable sequential listing **200** may preserve the order in which the at least a posted content took place by listing them in chronological order; alternatively or additionally, immutable sequential listing **200** may organize digitally signed assertions **204** into sub-listings **208** such as "blocks" in a blockchain, which may be themselves collected in a temporally sequential order; digitally signed assertions **204** within a sub-listing **208** may or may not be temporally sequential. The ledger may preserve the order in which at least a posted content took place by listing them in sub-listings **208** and placing the sub-listings **208** in chronological order. The immutable sequential listing **200** may be a distributed, consensus-based ledger, such as those operated according to the protocols promulgated by Ripple Labs, Inc., of San Francisco, Calif., or the Stellar Development Foundation, of San Francisco, Calif., or of Thunder Consensus. In some embodiments, the ledger is a secured ledger; in one embodiment, a secured ledger is a ledger having safeguards against alteration by unauthorized parties. The ledger may be maintained by a proprietor, such as a system administrator on a server, that controls access to the ledger; for instance, the user account controls may allow contributors to the ledger to add at least a posted content to the ledger, but may not allow any users to alter at least a posted content that have been added to the ledger. In some embodiments, ledger is cryptographically secured; in one embodiment, a ledger is cryptographically secured where each link in the chain contains encrypted or hashed information that makes it practically infeasible to alter the ledger without betraying that alteration has taken place, for instance by requiring that an administrator or other party sign new additions to the chain with a digital signature. Immutable sequential listing **200** may be incorporated in, stored in, or incorporate, any suitable data structure, including without limitation any database, datastore, file structure, distributed hash table, directed acyclic graph or the like. In some embodiments, the timestamp of an entry is cryptographically secured and validated via trusted time, either directly on the chain or indirectly by utilizing a separate chain. In one embodiment the validity of timestamp is provided using a time stamping authority as described in the RFC **3161** standard for trusted timestamps, or in the ANSI ASC x9.95 standard. In another embodiment, the trusted time ordering is provided by a group of entities collectively acting as the time stamping authority with a requirement that a threshold number of the group of authorities sign the timestamp.

In some embodiments, and with continued reference to FIG. **2**, immutable sequential listing **200**, once formed, may be inalterable by any party, no matter what access rights that party possesses. For instance, immutable sequential listing **200** may include a hash chain, in which data is added during a successive hashing process to ensure non-repudiation. Immutable sequential listing **200** may include a block chain. In one embodiment, a block chain is immutable sequential listing **200** that records one or more new at least a posted content in a data item known as a sub-listing **208** or "block." An example of a block chain is the BITCOIN block chain used to record BITCOIN transactions and values. Sub-listings **208** may be created in a way that places the sub-listings **208** in chronological order and link each sub-listing **208** to a previous sub-listing **208** in the chronological order so that any processor **104** may traverse the sub-listings **208** in reverse chronological order to verify any at least a posted content listed in the block chain. Each new sub-listing **208** may be required to contain a cryptographic hash describing the previous sub-listing **208**. In some embodiments, the block chain contains a single first sub-listing **208** sometimes known as a "genesis block."

Still referring to FIG. **2**, the creation of a new sub-listing **208** may be computationally expensive; for instance, the creation of a new sub-listing **208** may be designed by a "proof of work" protocol accepted by all participants in forming the immutable sequential listing **200** to take a powerful set of computing devices a certain period of time to produce. Where one sub-listing **208** takes less time for a given set of computing devices to produce the sub-listing **208** protocol may adjust the algorithm to produce the next sub-listing **208** so that it will require more steps; where one sub-listing **208** takes more time for a given set of computing devices to produce the sub-listing **208** protocol may adjust the algorithm to produce the next sub-listing **208** so that it will require fewer steps. As an example, protocol may require a new sub-listing **208** to contain a cryptographic hash describing its contents; the cryptographic hash may be required to satisfy a mathematical condition, achieved by having the sub-listing **208** contain a number, called a nonce, whose value is determined after the fact by the discovery of the hash that satisfies the mathematical condition. Continuing the example, the protocol may be able to adjust the mathematical condition so that the discovery of the hash describing a sub-listing **208** and satisfying the mathematical condition requires more or less steps, depending on the outcome of the previous hashing attempt. Mathematical condition, as an example, might be that the hash contains a certain number of leading zeros and a hashing algorithm that requires more steps to find a hash containing a greater number of leading zeros, and fewer steps to find a hash containing a lesser number of leading zeros. In some embodiments, production of a new sub-listing **208** according to the protocol is known as "mining." The creation of a new sub-listing **208** may be designed by a "proof of stake" protocol as will be apparent to those skilled in the art upon reviewing the entirety of this disclosure.

Continuing to refer to FIG. **2**, in some embodiments, protocol also creates an incentive to mine new sub-listings **208**. The incentive may be financial; for instance, successfully mining a new sub-listing **208** may result in the person or entity that mines the sub-listing **208** receiving a predetermined amount of currency. The currency may be fiat currency. Currency may be cryptocurrency as defined below. In other embodiments, incentive may be redeemed for particular products or services; the incentive may be a gift certificate with a particular business, for instance. In some embodiments, incentive is sufficiently attractive to cause participants to compete for the incentive by trying to race each other to the creation of sub-listings **208** Each sub-listing **208** created in immutable sequential listing **200** may contain a record or at least a posted content describing one or more addresses that receive an incentive, such as virtual currency, as the result of successfully mining the sub-listing **208**.

With continued reference to FIG. **2**, where two entities simultaneously create new sub-listings **208**, immutable sequential listing **200** may develop a fork; protocol may determine which of the two alternate branches in the fork is

the valid new portion of the immutable sequential listing **200** by evaluating, after a certain amount of time has passed, which branch is longer. "Length" may be measured according to the number of sub-listings **208** in the branch. Length may be measured according to the total computational cost of producing the branch. Protocol may treat only at least a posted content contained the valid branch as valid at least a posted content. When a branch is found invalid according to this protocol, at least a posted content registered in that branch may be recreated in a new sub-listing **208** in the valid branch; the protocol may reject "double spending" at least a posted content that transfer the same virtual currency that another at least a posted content in the valid branch has already transferred. As a result, in some embodiments the creation of fraudulent at least a posted content requires the creation of a longer immutable sequential listing **200** branch by the entity attempting the fraudulent at least a posted content than the branch being produced by the rest of the participants; as long as the entity creating the fraudulent at least a posted content is likely the only one with the incentive to create the branch containing the fraudulent at least a posted content, the computational cost of the creation of that branch may be practically infeasible, guaranteeing the validity of all at least a posted content in the immutable sequential listing **200**.

Still referring to FIG. **2**, additional data linked to at least a posted content may be incorporated in sub-listings **208** in the immutable sequential listing **200**; for instance, data may be incorporated in one or more fields recognized by block chain protocols that permit a person or computer forming a at least a posted content to insert additional data in the immutable sequential listing **200**. In some embodiments, additional data is incorporated in an unspendable at least a posted content field. For instance, the data may be incorporated in an OP_RETURN within the BITCOIN block chain. In other embodiments, additional data is incorporated in one signature of a multi-signature at least a posted content. In an embodiment, a multi-signature at least a posted content is at least a posted content to two or more addresses. In some embodiments, the two or more addresses are hashed together to form a single address, which is signed in the digital signature of the at least a posted content. In other embodiments, the two or more addresses are concatenated. In some embodiments, two or more addresses may be combined by a more complicated process, such as the creation of a Merkle tree or the like. In some embodiments, one or more addresses incorporated in the multi-signature at least a posted content are typical crypto-currency addresses, such as addresses linked to public keys as described above, while one or more additional addresses in the multi-signature at least a posted content contain additional data related to the at least a posted content; for instance, the additional data may indicate the purpose of the at least a posted content, aside from an exchange of virtual currency, such as the item for which the virtual currency was exchanged. In some embodiments, additional information may include network statistics for a given node of network, such as a distributed storage node, e.g. the latencies to nearest neighbors in a network graph, the identities or identifying information of neighboring nodes in the network graph, the trust level and/or mechanisms of trust (e.g. certificates of physical encryption keys, certificates of software encryption keys, (in non-limiting example certificates of software encryption may indicate the firmware version, manufacturer, hardware version and the like), certificates from a trusted third party, certificates from a decentralized anonymous authentication procedure, and other information quantifying the trusted status of the distributed storage node) of neighboring nodes in the network graph, IP addresses, GPS coordinates, and other information informing location of the node and/or neighboring nodes, geographically and/or within the network graph. In some embodiments, additional information may include history and/or statistics of neighboring nodes with which the node has interacted. In some embodiments, this additional information may be encoded directly, via a hash, hash tree or other encoding.

With continued reference to FIG. **2**, in some embodiments, virtual currency is traded as a crypto-currency. In one embodiment, a crypto-currency is a digital, currency such as Bitcoins, Peercoins, Namecoins, and Litecoins. Crypto-currency may be a clone of another crypto-currency. The crypto-currency may be an "alt-coin." Crypto-currency may be decentralized, with no particular entity controlling it; the integrity of the crypto-currency may be maintained by adherence by its participants to established protocols for exchange and for production of new currency, which may be enforced by software implementing the crypto-currency. Crypto-currency may be centralized, with its protocols enforced or hosted by a particular entity. For instance, crypto-currency may be maintained in a centralized ledger, as in the case of the XRP currency of Ripple Labs, Inc., of San Francisco, Calif. In lieu of a centrally controlling authority, such as a national bank, to manage currency values, the number of units of a particular crypto-currency may be limited; the rate at which units of crypto-currency enter the market may be managed by a mutually agreed-upon process, such as creating new units of currency when mathematical puzzles are solved, the degree of difficulty of the puzzles being adjustable to control the rate at which new units enter the market. Mathematical puzzles may be the same as the algorithms used to make productions of sub-listings **208** in a block chain computationally challenging; the incentive for producing sub-listings **208** may include the grant of new crypto-currency to the miners. Quantities of crypto-currency may be exchanged using at least a posted content as described above.

Turning now to FIG. **3**, an exemplary embodiment of a cryptographic accumulator **300** is illustrated. Cryptographic accumulator **300** has a plurality of accumulated elements **304**, each accumulated element **304** generated from a lot of the plurality of data lots. Accumulated elements **304** are create using an encryption process, defined for this purpose as a process that renders the lots of data unintelligible from the accumulated elements **304**; this may be a one-way process such as a cryptographic hashing process and/or a reversible process such as encryption. Cryptographic accumulator **300** further includes structures and/or processes for conversion of accumulated elements **304** to root **312** element. For instance, and as illustrated for exemplary purposes in FIG. **3**, cryptographic accumulator **300** may be implemented as a Merkle tree and/or hash tree, in which each accumulated element **304** created by cryptographically hashing a lot of data. Two or more accumulated elements **304** may be hashed together in a further cryptographic hashing process to produce a node **308** element; a plurality of node **308** elements may be hashed together to form parent nodes **308**, and ultimately a set of nodes **308** may be combined and cryptographically hashed to form root **312**. Contents of root **312** may thus be determined by contents of nodes **308** used to generate root **312**, and consequently by contents of accumulated elements **304**, which are determined by contents of lots used to generate accumulated elements **304**. As a result of collision resistance and avalanche effects of hashing algorithms, any change in any lot,

accumulated element **304**, and/or node **308** is virtually certain to cause a change in root **312**; thus, it may be computationally infeasible to modify any element of Merkle and/or hash tree without the modification being detectable as generating a different root **312**. In an embodiment, any accumulated element **304** and/or all intervening nodes **308** between accumulated element **304** and root **312** may be made available without revealing anything about a lot of data used to generate accumulated element **304**; lot of data may be kept secret and/or demonstrated with a secure proof as described below, preventing any unauthorized party from acquiring data in lot.

Alternatively or additionally, and still referring to FIG. **3**, cryptographic accumulator **300** may include a "vector commitment" which may act as an accumulator in which an order of elements in set is preserved in its root **312** and/or commitment. In an embodiment, a vector commitment may be a position binding commitment and can be opened at any position to a unique value with a short proof (sublinear in the length of the vector). A Merkle tree may be seen as a vector commitment with logarithmic size openings. Subvector commitments may include vector commitments where a subset of the vector positions can be opened in a single short proof (sublinear in the size of the subset). Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various alternative or additional cryptographic accumulators **300** that may be used as described herein. In addition to Merkle trees, accumulators may include without limitation RSA accumulators, class group accumulators, and/or bi-linear pairing-based accumulators. Any accumulator may operate using one-way functions that are easy to verify but infeasible to reverse, i.e. given an input it is easy to produce an output of the one-way function but given an output it is computationally infeasible and/or impossible to generate the input that produces the output via the one-way function. For instance, and by way of illustration, a Merkle tree may be based on a hash function as described above. Data elements may be hashed and grouped together. Then, the hashes of those groups may be hashed again and grouped together with the hashes of other groups; this hashing and grouping may continue until only a single hash remains. As a further non-limiting example, RSA and class group accumulators may be based on the fact that it is infeasible to compute an arbitrary root of an element in a cyclic group of unknown order, whereas arbitrary powers of elements are easy to compute. A data element may be added to the accumulator by hashing the data element successively until the hash is a prime number and then taking the accumulator to the power of that prime number. The witness may be the accumulator prior to exponentiation. Bi-linear paring-based accumulators may be based on the infeasibility found in elliptic curve cryptography, namely that finding a number k such that adding P to itself k times results in Q is impractical, whereas confirming that, given 4 points P, Q, R, S, the point, P needs to be added as many times to itself to result in Q as R needs to be added as many times to itself to result in S, can be computed efficiently for certain elliptic curves.

Referring now to FIG. **4**, an exemplary embodiment of a machine-learning module **400** that may perform one or more machine-learning processes as described in this disclosure is illustrated. Machine-learning module may perform determinations, classification, and/or analysis steps, methods, processes, or the like as described in this disclosure using machine learning processes. A "machine learning process," as previously defined, uses training data **404** to generate an algorithm that will be performed by a computing device/module to produce outputs **408** given data provided as inputs **412**; this is in contrast to a non-machine learning software program where the commands to be executed are determined in advance by a user and written in a programming language.

Still referring to FIG. **4**, "training data," as used herein, is data containing correlations that a machine-learning process may use to model relationships between two or more categories of data elements. For instance, and without limitation, training data **404** may include a plurality of data entries, each entry representing a set of data elements that were recorded, received, and/or generated together; data elements may be correlated by shared existence in a given data entry, by proximity in a given data entry, or the like. Multiple data entries in training data **404** may evince one or more trends in correlations between categories of data elements; for instance, and without limitation, a higher value of a first data element belonging to a first category of data element may tend to correlate to a higher value of a second data element belonging to a second category of data element, indicating a possible proportional or other mathematical relationship linking values belonging to the two categories. Multiple categories of data elements may be related in training data **404** according to various correlations; correlations may indicate causative and/or predictive links between categories of data elements, which may be modeled as relationships such as mathematical relationships by machine-learning processes as described in further detail below. Training data **404** may be formatted and/or organized by categories of data elements, for instance by associating data elements with one or more descriptors corresponding to categories of data elements. As a non-limiting example, training data **404** may include data entered in standardized forms by persons or processes, such that entry of a given data element in a given field in a form may be mapped to one or more descriptors of categories. Elements in training data **404** may be linked to descriptors of categories by tags, tokens, or other data elements; for instance, and without limitation, training data **404** may be provided in fixed-length formats, formats linking positions of data to categories such as comma-separated value (CSV) formats and/or self-describing formats such as extensible markup language (XML), JavaScript Object Notation (JSON), or the like, enabling processes or devices to detect categories of data.

Alternatively or additionally, and continuing to refer to FIG. **4**, training data **404** may include one or more elements that are not categorized; that is, training data **404** may not be formatted or contain descriptors for some elements of data. Machine-learning algorithms and/or other processes may sort training data **404** according to one or more categorizations using, for instance, natural language processing algorithms, tokenization, detection of correlated values in raw data and the like; categories may be generated using correlation and/or other processing algorithms. As a non-limiting example, in a corpus of text, phrases making up a number "n" of compound words, such as nouns modified by other nouns, may be identified according to a statistically significant prevalence of n-grams containing such words in a particular order; such an n-gram may be categorized as an element of language such as a "word" to be tracked similarly to single words, generating a new category as a result of statistical analysis. Similarly, in a data entry including some textual data, a person's name may be identified by reference to a list, dictionary, or other compendium of terms, permitting ad-hoc categorization by machine-learning algorithms, and/or automated association of data in the data entry with descriptors or into a given format. The ability to categorize data entries automatedly may enable the same training data **404** to be made applicable for two or more distinct machine-

learning algorithms as described in further detail below. Training data **404** used by machine-learning module **400** may correlate any input data as described in this disclosure to any output data as described in this disclosure

Further referring to FIG. **4**, training data may be filtered, sorted, and/or selected using one or more supervised and/or unsupervised machine-learning processes and/or models as described in further detail below; such models may include without limitation a training data classifier **416**. Training data classifier **416** may include a "classifier," which as used in this disclosure is a machine-learning model as defined below, such as a mathematical model, neural net, or program generated by a machine learning algorithm known as a "classification algorithm," as described in further detail below, that sorts inputs into categories or bins of data, outputting the categories or bins of data and/or labels associated therewith. A classifier may be configured to output at least a datum that labels or otherwise identifies a set of data that are clustered together, found to be close under a distance metric as described below, or the like. Machine-learning module **400** may generate a classifier using a classification algorithm, defined as a processes whereby a computing device and/or any module and/or component operating thereon derives a classifier from training data **404**. Classification may be performed using, without limitation, linear classifiers such as without limitation logistic regression and/or naive Bayes classifiers, nearest neighbor classifiers such as k-nearest neighbors classifiers, support vector machines, least squares support vector machines, fisher's linear discriminant, quadratic classifiers, decision trees, boosted trees, random forest classifiers, learning vector quantization, and/or neural network-based classifiers

Still referring to FIG. **4**, machine-learning module **400** may be configured to perform a lazy-learning process **420** and/or protocol, which may alternatively be referred to as a "lazy loading" or "call-when-needed" process and/or protocol, may be a process whereby machine learning is conducted upon receipt of an input to be converted to an output, by combining the input and training set to derive the algorithm to be used to produce the output on demand. For instance, an initial set of simulations may be performed to cover an initial heuristic and/or "first guess" at an output and/or relationship. As a non-limiting example, an initial heuristic may include a ranking of associations between inputs and elements of training data **404**. Heuristic may include selecting some number of highest-ranking associations and/or training data **404** elements. Lazy learning may implement any suitable lazy learning algorithm, including without limitation a K-nearest neighbors algorithm, a lazy naive Bayes algorithm, or the like; persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various lazy-learning algorithms that may be applied to generate outputs as described in this disclosure, including without limitation lazy learning applications of machine-learning algorithms.

Alternatively or additionally, and with continued reference to FIG. **4**, machine-learning processes as described in this disclosure may be used to generate machine-learning models **424**. A "machine-learning model," as used in this disclosure, is a mathematical and/or algorithmic representation of a relationship between inputs and outputs, as generated using any machine-learning process including without limitation any process as described above, and stored in memory; an input is submitted to a machine-learning model **424** once created, which generates an output based on the relationship that was derived. For instance, and without limitation, a linear regression model, generated using a linear regression algorithm, may compute a linear combination of input data using coefficients derived during machine-learning processes to calculate an output datum. As a further non-limiting example, a machine-learning model **424** may be generated by creating an artificial neural network, such as a convolutional neural network comprising an input layer of nodes, one or more intermediate layers, and an output layer of nodes. Connections between nodes may be created via the process of "training" the network, in which elements from a training data **404** set are applied to the input nodes, a suitable training algorithm (such as Levenberg-Marquardt, conjugate gradient, simulated annealing, or other algorithms) is then used to adjust the connections and weights between nodes in adjacent layers of the neural network to produce the desired values at the output nodes. This process is sometimes referred to as deep learning.

Still referring to FIG. **4**, machine-learning algorithms may include at least a supervised machine-learning process **428**. At least a supervised machine-learning process **428**, as defined herein, include algorithms that receive a training set relating a number of inputs to a number of outputs, and seek to find one or more mathematical relations relating inputs to outputs, where each of the one or more mathematical relations is optimal according to some criterion specified to the algorithm using some scoring function. For instance, a supervised learning algorithm may include and input and outputs described in this disclosure, and a scoring function representing a desired form of relationship to be detected between inputs and outputs; scoring function may, for instance, seek to maximize the probability that a given input and/or combination of elements inputs is associated with a given output to minimize the probability that a given input is not associated with a given output. Scoring function may be expressed as a risk function representing an "expected loss" of an algorithm relating inputs to outputs, where loss is computed as an error function representing a degree to which a prediction generated by the relation is incorrect when compared to a given input-output pair provided in training data **404**. Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various possible variations of at least a supervised machine-learning process **428** that may be used to determine relation between inputs and outputs. Supervised machine-learning processes may include classification algorithms as defined above.

Further referring to FIG. **4**, machine learning processes may include at least an unsupervised machine-learning processes **432**. An unsupervised machine-learning process, as used herein, is a process that derives inferences in datasets without regard to labels; as a result, an unsupervised machine-learning process may be free to discover any structure, relationship, and/or correlation provided in the data. Unsupervised processes may not require a response variable; unsupervised processes may be used to find interesting patterns and/or inferences between variables, to determine a degree of correlation between two or more variables, or the like.

Still referring to FIG. **4**, machine-learning module **400** may be designed and configured to create a machine-learning model **424** using techniques for development of linear regression models. Linear regression models may include ordinary least squares regression, which aims to minimize the square of the difference between predicted outcomes and actual outcomes according to an appropriate norm for measuring such a difference (e.g. a vector-space distance norm); coefficients of the resulting linear equation may be modified to improve minimization. Linear regres-

sion models may include ridge regression methods, where the function to be minimized includes the least-squares function plus term multiplying the square of each coefficient by a scalar amount to penalize large coefficients. Linear regression models may include least absolute shrinkage and selection operator (LASSO) models, in which ridge regression is combined with multiplying the least-squares term by a factor of 1 divided by double the number of samples. Linear regression models may include a multi-task lasso model wherein the norm applied in the least-squares term of the lasso model is the Frobenius norm amounting to the square root of the sum of squares of all terms. Linear regression models may include the elastic net model, a multi-task elastic net model, a least angle regression model, a LARS lasso model, an orthogonal matching pursuit model, a Bayesian regression model, a logistic regression model, a stochastic gradient descent model, a perceptron model, a passive aggressive algorithm, a robustness regression model, a Huber regression model, or any other suitable model that may occur to persons skilled in the art upon reviewing the entirety of this disclosure. Linear regression models may be generalized in an embodiment to polynomial regression models, whereby a polynomial equation (e.g. a quadratic, cubic or higher-order equation) providing a best predicted output/actual output fit is sought; similar methods to those described above may be applied to minimize error functions, as will be apparent to persons skilled in the art upon reviewing the entirety of this disclosure.

Continuing to refer to FIG. **4**, machine-learning algorithms may include, without limitation, linear discriminant analysis. Machine-learning algorithm may include quadratic discriminate analysis. Machine-learning algorithms may include kernel ridge regression. Machine-learning algorithms may include support vector machines, including without limitation support vector classification-based regression processes. Machine-learning algorithms may include stochastic gradient descent algorithms, including classification and regression algorithms based on stochastic gradient descent. Machine-learning algorithms may include nearest neighbors algorithms. Machine-learning algorithms may include various forms of latent space regularization such as variational regularization. Machine-learning algorithms may include Gaussian processes such as Gaussian Process Regression. Machine-learning algorithms may include cross-decomposition algorithms, including partial least squares and/or canonical correlation analysis. Machine-learning algorithms may include naive Bayes methods. Machine-learning algorithms may include algorithms based on decision trees, such as decision tree classification or regression algorithms. Machine-learning algorithms may include ensemble methods such as bagging meta-estimator, forest of randomized tress, AdaBoost, gradient tree boosting, and/or voting classifier methods. Machine-learning algorithms may include neural net algorithms, including convolutional neural net processes.

Referring now to FIG. **5**, an exemplary embodiment of neural network **500** is illustrated. A neural network **500** also known as an artificial neural network, is a network of "nodes," or data structures having one or more inputs, one or more outputs, and a function determining outputs based on inputs. Such nodes may be organized in a network, such as without limitation a convolutional neural network, including an input layer of nodes **504**, one or more intermediate layers **508**, and an output layer of nodes **512**. Connections between nodes may be created via the process of "training" the network, in which elements from a training dataset are applied to the input nodes, a suitable training algorithm (such as Levenberg-Marquardt, conjugate gradient, simulated annealing, or other algorithms) is then used to adjust the connections and weights between nodes in adjacent layers of the neural network to produce the desired values at the output nodes. This process is sometimes referred to as deep learning. Connections may run solely from input nodes toward output nodes in a "feed-forward" network, or may feed outputs of one layer back to inputs of the same or a different layer in a "recurrent network."

Referring now to FIG. **6**, an exemplary embodiment of a node of a neural network is illustrated. A node may include, without limitation a plurality of inputs x_i that may receive numerical values from inputs to a neural network containing the node and/or from other nodes. Node may perform a weighted sum of inputs using weights w_i that are multiplied by respective inputs x_i. Additionally or alternatively, a bias b may be added to the weighted sum of the inputs such that an offset is added to each unit in the neural network layer that is independent of the input to the layer. The weighted sum may then be input into a function φ, which may generate one or more outputs y. Weight w_i applied to an input x_i may indicate whether the input is "excitatory," indicating that it has strong influence on the one or more outputs y, for instance by the corresponding weight having a large numerical value, and/or a "inhibitory," indicating it has a weak effect influence on the one more inputs y, for instance by the corresponding weight having a small numerical value. The values of weights w_i may be determined by training a neural network using training data, which may be performed using any suitable process as described above.

Referring to FIG. **7**, an exemplary embodiment of fuzzy set comparison **700** is illustrated. A first fuzzy set **704** may be represented, without limitation, according to a first membership function **708** representing a probability that an input falling on a first range of values **712** is a member of the first fuzzy set **704**, where the first membership function **708** has values on a range of probabilities such as without limitation the interval [0,1], and an area beneath the first membership function **708** may represent a set of values within first fuzzy set **704**. Although first range of values **712** is illustrated for clarity in this exemplary depiction as a range on a single number line or axis, first range of values **712** may be defined on two or more dimensions, representing, for instance, a Cartesian product between a plurality of ranges, curves, axes, spaces, dimensions, or the like. First membership function **708** may include any suitable function mapping first range **712** to a probability interval, including without limitation a triangular function defined by two linear elements such as line segments or planes that intersect at or below the top of the probability interval. As a non-limiting example, triangular membership function may be defined as:

$$y(x, a, b, c) = \begin{cases} 0, \text{ for } x > c \text{ and } x < a \\ \dfrac{x-a}{b-a}, \text{ for } a \leq x < b \\ \dfrac{c-x}{c-b}, \text{ if } b < x \leq c \end{cases}$$

a trapezoidal membership function may be defined as:

$$y(x, a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

a sigmoidal function may be defined as:

$$y(x, a, c) = \frac{1}{1 - e^{-a(x-c)}}$$

a Gaussian membership function may be defined as:

$$y(x, c, \sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2}$$

and a bell membership function may be defined as:

$$y(x, a, b, c,) = \left[1 + \left|\frac{x-c}{a}\right|^{2b}\right]^{-1}$$

Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various alternative or additional membership functions that may be used consistently with this disclosure.

Still referring to FIG. **7**, first fuzzy set **704** may represent any value or combination of values as described above, including output from one or more machine-learning models and lost user data **120**, a predetermined class, such as without limitation, previously verified user data. A second fuzzy set **716**, which may represent any value which may be represented by first fuzzy set **704**, may be defined by a second membership function **720** on a second range **724**; second range **724** may be identical and/or overlap with first range **712** and/or may be combined with first range via Cartesian product or the like to generate a mapping permitting evaluation overlap of first fuzzy set **704** and second fuzzy set **716**. Where first fuzzy set **704** and second fuzzy set **716** have a region **728** that overlaps, first membership function **708** and second membership function **720** may intersect at a point **732** representing a probability, as defined on probability interval, of a match between first fuzzy set **704** and second fuzzy set **716**. Alternatively or additionally, a single value of first and/or second fuzzy set may be located at a locus **736** on first range **712** and/or second range **724**, where a probability of membership may be taken by evaluation of first membership function **708** and/or second membership function **720** at that range point. A probability at **728** and/or **732** may be compared to a threshold **740** to determine whether a positive match is indicated. Threshold **740** may, in a non-limiting example, represent a degree of match between first fuzzy set **704** and second fuzzy set **716**, and/or single values therein with each other or with either set, which is sufficient for purposes of the matching process; for instance, threshold may indicate a sufficient degree of overlap between an output from one or more machine-learning models and/or lost user data **120** and a predetermined class, such as without limitation previously verified user data, for combination to occur as described above. Alternatively or additionally, each threshold may be tuned by a machine-learning and/or statistical process, for instance and without limitation as described in further detail below.

Further referring to FIG. **7**, in an embodiment, a degree of match between fuzzy sets may be used to classify lost user data **120** with previously verified user data. For instance, if lost user data **120** has a fuzzy set matching previously verified user data fuzzy set by having a degree of overlap exceeding a threshold, processor **104** and/or computing device **112** may classify lost user data **120** as belonging to the previously verified user data. Where multiple fuzzy matches are performed, degrees of match for each respective fuzzy set may be computed and aggregated through, for instance, addition, averaging, or the like, to determine an overall degree of match.

Still referring to FIG. **7**, in an embodiment, lost user data **120** may be compared to multiple previously verified user data fuzzy sets. For instance, lost user data **120** may be represented by a fuzzy set that is compared to each of the multiple previously verified user data fuzzy sets; and a degree of overlap exceeding a threshold between lost user data **120** fuzzy set and any of the multiple previously verified user data fuzzy sets may cause processor **104** and/or computing device **112** to classify lost user data **120** as belonging to previously verified user data. For instance, in one embodiment there may be two previously verified user data fuzzy sets, representing respectively first previously verified user data and second previously verified user data. First previously verified user data may have a first fuzzy set; Second previously verified user data may have a second fuzzy set; and lost user data **120** may have lost user data **120** fuzzy set. Processor **104** and/or computing device **112**, for example, may compare lost user data **120** fuzzy set with each of previously verified user data fuzzy set and previously verified user data fuzzy set, as described above, and classify lost user data **120** to either, both, or neither of first previously verified user data or second previously verified user data. Machine-learning methods as described throughout may, in a non-limiting example, generate coefficients used in fuzzy set equations as described above, such as without limitation x, c, and σ of a Gaussian set as described above, as outputs of machine-learning methods. Likewise, lost user data **120** may be used indirectly to determine a fuzzy set, as lost user data **120** fuzzy set may be derived from outputs of one or more machine-learning models that take lost user data **120** directly or indirectly as inputs.

Referring now to FIG. **8**, is a flow diagram illustrating an exemplary embodiment of a method **800** for using a computing device to verify lost user data. The computing device may be any computing device described throughout this disclosure, for example and with reference to FIG. **1**. At step **805**, method **800** includes using a computing device configured to identify a plurality of lost user data stored in a resource data storage system potentially associated with a particular user of a plurality of users. The resource data storage system is stored on an immutable sequential listing and identifying a plurality of lost user data includes dividing the plurality of lost user data into a plurality of user identifier subsets that are identified by at least a public key. After which, each of the plurality of user identifier subsets is matched with the particular user of the plurality of users. This may be implemented as disclosed with reference to FIGS. **1-7**. A user may have multiple user identifier subsets which may be implemented, for instance, as immutable sequential listing addresses, public keys of public-key cryptographic systems, or the like. In some embodiments, dividing lost user data into user identifier subsets may include using machine learning process such any classification algorithm described in FIGS. **1** and **4** to output data bins of matched lost user data to a particular user. Lost user data may be a data record that could not be verified and properly added to the resource data storage system at the time of its creation by a user. In an embodiment, information stored in the resource storage system may include user-specific historical data, other job information, education information, or the like. The resource data storage system is be stored on an immutable sequential listing as described with reference to

FIGS. **1-7**. The resource data storage system may contain entries representing data of at least a job candidate, additionally, entries of data may include files, such as JPEGs, documents, spreadsheets, videos, pictures, etc., as described in FIG. **1**. In some embodiments, the immutable sequential listing may contain a cryptographic accumulator, as disclosed and with reference to FIGS. **1** and **3**.

In some cases, a data record in an immutable sequential listing may include a user's resume or profile, which may be in video, visual, audio and/or text format, a cover letter and other information relating to the applicant's education, experience (professional, work, personal), qualifications, interests, and the like, among others. Sometimes separate blocks can be produced concurrently, creating a temporary fork in the immutable sequential listing. Blocks not selected for inclusion in the chain are called orphan blocks. In some embodiments, the plurality of lost user data may include a plurality of orphan blocks in a cryptographic hash chain existing outside of a main hash chain on the immutable sequential listing. In some embodiments the computing device may be configured to run as a full node to monitor the creation of blocks to identify lost user data as described in FIG. **1**.

Still referring to FIG. **8**, at step **810**, method **800** includes using a computing device configured to verify, using a recovery program, a plurality of lost user data potentially associated with a particular user. Method of verification may be method described through this disclosure, for example and with reference to FIGS. **1-7**. In some embodiments, the recovery program, as defined in FIG. **1**, may include sending a user an electronic request, such as an automated email, for verification using public-key cryptography, as disclosed above, wherein the user identifier subsets associated with a public key which may be known to others) may be recovered by a user digitally signing with a private key (which may not be known by anyone except the user). In an embodiment, the user may get a verification request on another computing device known to be owned by the user. In some embodiments, verification may include proof of possession, as described in FIG. **1**, wherein a digital signature may be signed with a private key such that decrypting the block with the public key proves possession. In some embodiments, verifying a plurality of lost user data may include using a cryptographic commitment, for example and with reference to FIG. **1**. In some embodiments, verifying a plurality of lost user data may include using machine a learning process such as a classifier and fuzzy comparison, for example and with reference to FIG. **1-7**.

Still referring to FIG. **8**, at step **815**, method **800** includes using a computing device configured to link a plurality of lost user data to a particular user. In some embodiments, linking may include cryptographically inserting into the main hash-chain on the immutable sequential listing, the plurality of lost user data associated with the particular user, for example and with reference to FIGS. **1** and **3**.

It is to be noted that any one or more of the aspects and embodiments described herein may be conveniently implemented using one or more machines (e.g., one or more computing devices that are utilized as a user computing device for an electronic document, one or more server devices, such as a document server, etc.) programmed according to the teachings of the present specification, as will be apparent to those of ordinary skill in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those of ordinary skill in the software art. Aspects and implementations discussed above employing software and/or software modules may also include appropriate hardware for assisting in the implementation of the machine executable instructions of the software and/or software module.

Such software may be a computer program product that employs a machine-readable storage medium. A machine-readable storage medium may be any medium that is capable of storing and/or encoding a sequence of instructions for execution by a machine (e.g., a computing device) and that causes the machine to perform any one of the methodologies and/or embodiments described herein. Examples of a machine-readable storage medium include, but are not limited to, a magnetic disk, an optical disc (e.g., CD, CD-R, DVD, DVD-R, etc.), a magneto-optical disk, a read-only memory "ROM" device, a random access memory "RAM" device, a magnetic card, an optical card, a solid-state memory device, an EPROM, an EEPROM, and any combinations thereof. A machine-readable medium, as used herein, is intended to include a single medium as well as a collection of physically separate media, such as, for example, a collection of compact discs or one or more hard disk drives in combination with a computer memory. As used herein, a machine-readable storage medium does not include transitory forms of signal transmission.

Such software may also include information (e.g., data) carried as a data signal on a data carrier, such as a carrier wave. For example, machine-executable information may be included as a data-carrying signal embodied in a data carrier in which the signal encodes a sequence of instruction, or portion thereof, for execution by a machine (e.g., a computing device) and any related information (e.g., data structures and data) that causes the machine to perform any one of the methodologies and/or embodiments described herein.

Examples of a computing device include, but are not limited to, an electronic book reading device, a computer workstation, a terminal computer, a server computer, a handheld device (e.g., a tablet computer, a smartphone, etc.), a web appliance, a network router, a network switch, a network bridge, any machine capable of executing a sequence of instructions that specify an action to be taken by that machine, and any combinations thereof. In one example, a computing device may include and/or be included in a kiosk.

FIG. **9** shows a diagrammatic representation of one embodiment of a computing device in the exemplary form of a computer system **900** within which a set of instructions for causing a control system to perform any one or more of the aspects and/or methodologies of the present disclosure may be executed. It is also contemplated that multiple computing devices may be utilized to implement a specially configured set of instructions for causing one or more of the devices to perform any one or more of the aspects and/or methodologies of the present disclosure. Computer system **900** includes a processor **904** and a memory **908** that communicate with each other, and with other components, via a bus **912**. Bus **912** may include any of several types of bus structures including, but not limited to, a memory bus, a memory controller, a peripheral bus, a local bus, and any combinations thereof, using any of a variety of bus architectures.

Processor **904** may include any suitable processor, such as without limitation a processor incorporating logical circuitry for performing arithmetic and logical operations, such as an arithmetic and logic unit (ALU), which may be regulated with a state machine and directed by operational inputs from memory and/or sensors; processor **904** may be organized according to Von Neumann and/or Harvard architecture as a

non-limiting example. Processor **904** may include, incorporate, and/or be incorporated in, without limitation, a microcontroller, microprocessor, digital signal processor (DSP), Field Programmable Gate Array (FPGA), Complex Programmable Logic Device (CPLD), Graphical Processing Unit (GPU), general purpose GPU, Tensor Processing Unit (TPU), analog or mixed signal processor, Trusted Platform Module (TPM), a floating point unit (FPU), and/or system on a chip (SoC).

Memory **908** may include various components (e.g., machine-readable media) including, but not limited to, a random-access memory component, a read only component, and any combinations thereof. In one example, a basic input/output system **916** (BIOS), including basic routines that help to transfer information between elements within computer system **900**, such as during start-up, may be stored in memory **908**. Memory **908** may also include (e.g., stored on one or more machine-readable media) instructions (e.g., software) **920** embodying any one or more of the aspects and/or methodologies of the present disclosure. In another example, memory **908** may further include any number of program modules including, but not limited to, an operating system, one or more application programs, other program modules, program data, and any combinations thereof.

Computer system **900** may also include a storage device **924**. Examples of a storage device (e.g., storage device **924**) include, but are not limited to, a hard disk drive, a magnetic disk drive, an optical disc drive in combination with an optical medium, a solid-state memory device, and any combinations thereof. Storage device **924** may be connected to bus **912** by an appropriate interface (not shown). Example interfaces include, but are not limited to, SCSI, advanced technology attachment (ATA), serial ATA, universal serial bus (USB), IEEE 1394 (FIREWIRE), and any combinations thereof. In one example, storage device **924** (or one or more components thereof) may be removably interfaced with computer system **900** (e.g., via an external port connector (not shown)). Particularly, storage device **924** and an associated machine-readable medium **928** may provide nonvolatile and/or volatile storage of machine-readable instructions, data structures, program modules, and/or other data for computer system **900**. In one example, software **920** may reside, completely or partially, within machine-readable medium **928**. In another example, software **920** may reside, completely or partially, within processor **904**.

Computer system **900** may also include an input device **932**. In one example, a user of computer system **900** may enter commands and/or other information into computer system **900** via input device **932**. Examples of an input device **932** include, but are not limited to, an alpha-numeric input device (e.g., a keyboard), a pointing device, a joystick, a gamepad, an audio input device (e.g., a microphone, a voice response system, etc.), a cursor control device (e.g., a mouse), a touchpad, an optical scanner, a video capture device (e.g., a still camera, a video camera), a touchscreen, and any combinations thereof. Input device **932** may be interfaced to bus **912** via any of a variety of interfaces (not shown) including, but not limited to, a serial interface, a parallel interface, a game port, a USB interface, a FIREWIRE interface, a direct interface to bus **912**, and any combinations thereof. Input device **932** may include a touch screen interface that may be a part of or separate from display **936**, discussed further below. Input device **932** may be utilized as a user selection device for selecting one or more graphical representations in a graphical interface as described above.

A user may also input commands and/or other information to computer system **900** via storage device **924** (e.g., a removable disk drive, a flash drive, etc.) and/or network interface device **940**. A network interface device, such as network interface device **940**, may be utilized for connecting computer system **900** to one or more of a variety of networks, such as network **944**, and one or more remote devices **948** connected thereto. Examples of a network interface device include, but are not limited to, a network interface card (e.g., a mobile network interface card, a LAN card), a modem, and any combination thereof. Examples of a network include, but are not limited to, a wide area network (e.g., the Internet, an enterprise network), a local area network (e.g., a network associated with an office, a building, a campus or other relatively small geographic space), a telephone network, a data network associated with a telephone/voice provider (e.g., a mobile communications provider data and/or voice network), a direct connection between two computing devices, and any combinations thereof. A network, such as network **944**, may employ a wired and/or a wireless mode of communication. In general, any network topology may be used. Information (e.g., data, software **920**, etc.) may be communicated to and/or from computer system **900** via network interface device **940**.

Computer system **900** may further include a video display adapter **952** for communicating a displayable image to a display device, such as display device **936**. Examples of a display device include, but are not limited to, a liquid crystal display (LCD), a cathode ray tube (CRT), a plasma display, a light emitting diode (LED) display, and any combinations thereof. Display adapter **952** and display device **936** may be utilized in combination with processor **904** to provide graphical representations of aspects of the present disclosure. In addition to a display device, computer system **900** may include one or more other peripheral output devices including, but not limited to, an audio speaker, a printer, and any combinations thereof. Such peripheral output devices may be connected to bus **912** via a peripheral interface **956**. Examples of a peripheral interface include, but are not limited to, a serial port, a USB connection, a FIREWIRE connection, a parallel connection, and any combinations thereof

The foregoing has been a detailed description of illustrative embodiments of the invention. Various modifications and additions can be made without departing from the spirit and scope of this invention. Features of each of the various embodiments described above may be combined with features of other described embodiments as appropriate in order to provide a multiplicity of feature combinations in associated new embodiments. Furthermore, while the foregoing describes a number of separate embodiments, what has been described herein is merely illustrative of the application of the principles of the present invention. Additionally, although particular methods herein may be illustrated and/or described as being performed in a specific order, the ordering is highly variable within ordinary skill to achieve apparatuses, systems, and software according to the present disclosure. Accordingly, this description is meant to be taken only by way of example, and not to otherwise limit the scope of this invention.

Exemplary embodiments have been disclosed above and illustrated in the accompanying drawings. It will be understood by those skilled in the art that various changes, omissions and additions may be made to that which is specifically disclosed herein without departing from the spirit and scope of the present invention.

What is claimed is:

1. An apparatus for using machine learning to verify lost user data in a resource data storage system, the apparatus comprising:

at least a processor; and

a memory communicatively connected to the processor, the memory containing instructions configuring the at least a processor to:

identify a plurality of lost user data stored in a resource data storage system potentially associated with a particular user of a plurality of users, the resource data storage system stored on an immutable sequential listing, wherein identifying a plurality of lost user data comprises dividing the plurality of lost user data into a plurality of user identifier subsets that are identified by at least a public key, wherein each of the plurality of user identifier subsets is matched with the particular user of the plurality of users;

verify, using a recovery program, the plurality of lost user data potentially associated with the particular user and;

link, as a function of the verification, the plurality of lost user data to the particular user.

2. The apparatus of claim 1, wherein the immutable sequential listing includes a cryptographic accumulator.

3. The apparatus of claim 1, wherein the plurality of lost user data comprises a plurality of orphan blocks in a cryptographic hash chain existing outside of a main hash chain on the immutable sequential listing.

4. The apparatus of claim 1, wherein dividing the plurality of lost user data into user identifier subsets comprises using a machine learning process to output data bins of matched lost user data to a particular user.

5. The apparatus of claim 4, wherein the machine learning process includes using a classification algorithm.

6. The apparatus of claim 1, wherein the recovery program comprises using public-key cryptography.

7. The apparatus of claim 1, wherein verifying the plurality of lost user data comprises using fuzzy set comparison.

8. The apparatus of claim 1, wherein verifying the plurality of lost user data comprises using proof of possession.

9. The apparatus of claim 8, wherein verifying the plurality of lost user data comprises using a cryptographic commitment.

10. The apparatus of claim 3, wherein linking the plurality of lost user data further comprises cryptographically inserting into the main hash-chain on the immutable sequential listing, the plurality of lost user data associated with the particular user.

11. A method for using machine learning to verify lost user data in a resource data storage system, the method comprising:

identifying, using a computing device, a plurality of lost user data stored in a resource data storage system potentially associated with a particular user of a plurality of users, the resource data storage system stored on an immutable sequential listing, wherein identifying a plurality of lost user data comprises dividing the plurality of lost user data into a plurality of user identifier subsets that are identified by at least a public key, wherein each of the plurality of user identifier subsets is matched with the particular user of the plurality of users;

verifying, using a recovery program and a computing device, the plurality of lost user data potentially associated with the particular user and;

linking, using a computing device, as a function of the verification, the plurality of lost user data to the particular user.

12. The method of claim 11, wherein the immutable sequential listing includes a cryptographic accumulator.

13. The method of claim 11, wherein the plurality of lost user data comprises a plurality of orphan blocks in a cryptographic hash chain existing outside of a main hash chain on the immutable sequential listing.

14. The method of claim 11, wherein dividing the plurality of lost user data into user identifier subsets comprises using a machine learning process to output data bins of matched lost user data to a particular user.

15. The method of claim 14, wherein the machine learning process includes using a classification algorithm.

16. The method of claim 11, wherein the recovery program comprises using public-key cryptography.

17. The method of claim 11, wherein verifying the plurality of lost user data comprises using fuzzy set comparison.

18. The method of claim 11, wherein verifying the plurality of lost user data comprises using proof of possession.

19. The method of claim 18, wherein verifying the plurality of lost user data comprises using a cryptographic commitment.

20. The method of claim 13, wherein linking the plurality of lost user data further comprises cryptographically inserting into the main hash-chain on the immutable sequential listing, the plurality of lost user data associated with the particular user.

* * * * *